

**REVIEW**

510-4DDS/0197  
CSC 10038085

**MISSION OPERATIONS AND DATA SYSTEMS DIRECTORATE**

**Earth Science  
Data and Information System (ESDIS)  
Level 1 Product Generation System (LPGS)  
Detailed Design Specification (DDS)**

**August 1997**



National Aeronautics and  
Space Administration

\_\_\_\_\_  
Goddard Space Flight Center  
Greenbelt, Maryland

## REVIEW

# Earth Science Data and Information System (ESDIS) Level 1 Product Generation System (LPGS) Detailed Design Specification (DDS)

August 1997

Prepared Under Contract NAS5-31000/HQ001057  
By Computer Sciences Corporation  
CSC 10038085

### Prepared by:

\_\_\_\_\_  
B. Pederson Date  
Software Engineer, CSC

### Quality Assured by:

\_\_\_\_\_  
S. Whisonant Date  
Quality Assurance Officer,  
Landsat 7 Project, CSC

### Reviewed by:

\_\_\_\_\_  
D. Derrick Date  
LPGS Technical Manager, CSC

### Approved by:

\_\_\_\_\_  
M. Samii Date  
Landsat 7 Technical Area Manager, CSC

\_\_\_\_\_  
D. Badolato Date  
LPGS Software Development Manager,  
GSFC, Code 551

\_\_\_\_\_  
J. Henegar Date  
LPGS Project Manager,  
GSFC, Code 514

\_\_\_\_\_  
J. Pizzola  
LPGS Project Manager, CSC

**Goddard Space Flight Center**  
Greenbelt, Maryland

**REVIEW**

## **Preface**

---

## REVIEW

# Abstract

---

The Earth Science Data and Information System (ESDIS) Level 1 Product Generation System (LPGS) will be operated within the Earth Observing System (EOS) Ground System (EGS) to provide Landsat 7 Enhanced Thematic Mapper Plus (ETM+) systematically corrected digital images for distribution to the Earth Observing System (EOS) Data and Information System (EOSDIS) Core System (ECS) customers. The software detailed design presented in this document is based on the requirements contained in the LPGS Functional and Performance Requirements Specification (F&PRS), the LPGS Operations Concept document, and the LPGS System Design Specification (SDS).

**Keywords:** *Earth Observing System (EOS) Ground System (EGS), Earth Science Data and Information System (ESDIS), Landsat 7, Level 1 Product Generation System (LPGS)*

## REVIEW

### Change Information Page

List of Effective Pages			
Page Number		Issue	
Title page		Original	
iii through xxi		Original	
1-1 through 1-5		Original	
2-1 through 2-15		Original	
3-1 through 3-9		Original	
4-1 through 4-9		Original	
5-1 through 5-38		Original	
6-1 through 6-88		Original	
7-1 through 7-2		Original	
8-1		Original	
9-1 through 9-12		Original	
10-1 through 10-11		Original	
11-1 through 11-11		Original	
12-1 through 12-6		Original	
A-1 through A-13		Original	
B-1		Original	
C-1		Original	
AB-1 through AB-6		Original	
Document History			
Document Number	Status/Issue	Publication Date	CCR Number
510-4DDS/0197 CSC 10038085	Original	August 1997	NA

# REVIEW

## Contents

---

### Preface

### Abstract

## Section 1. Introduction

1.1	Purpose .....	1-1
1.2	Scope .....	1-1
1.3	Applicable Documents .....	1-1
1.3.1	Requirements Documents.....	1-1
1.3.2	Reference Documents .....	1-2
1.4	Definitions .....	1-3
1.5	Document Organization.....	1-5

## Section 2. Detailed Design Overview

2.1	Design Methodology .....	2-1
2.1.1	Design Process.....	2-1
2.1.2	Design Products .....	2-1
2.1.3	Design Conventions .....	2-2
2.2	LPGS General Overview .....	2-2
2.2.1	Hardware .....	2-6
2.2.1.1	LPGS System Hardware .....	2-6
2.2.2	Software .....	2-9
2.2.2.1	LPGS Application Software.....	2-9

## REVIEW

2.2.2.2	LPGS Support Software .....	2-12
2.2.3	External Interfaces .....	2-12
2.2.3.1	Interface With the ECS .....	2-13
2.2.3.2	Interface With the IAS.....	2-13
2.3	Design Considerations .....	2-13
2.3.1	Global Library Areas .....	2-13
2.3.2	Database Interface.....	2-13
2.3.3	User Interface .....	2-13
2.3.4	Interprocess Communication.....	2-14
2.3.4.1	Database Communication .....	2-14
2.3.4.2	Directives .....	2-14
2.3.4.3	Oracle Pipes.....	2-14
2.3.5	Error Handling Philosophy.....	2-14
2.3.5.1	Error Reporting .....	2-14
2.3.5.2	Error Processing .....	2-15
2.3.6	File Maintenance.....	2-15

## Section 3. LPGS Operational Scenarios

3.1	Introduction.....	3-1
3.2	Operations Staff .....	3-1
3.3	Nominal Operations .....	3-2
3.3.1	Start Up LPGS .....	3-2
3.3.2	Shut Down LPGS .....	3-2
3.3.3	Process L1 Product (Nominal End-to-End Processing Flow).....	3-3
3.3.4	Cancel L1 Processing.....	3-5

## REVIEW

3.3.5	Retrieval of Characterization Results by IAS .....	3-6
3.4	Non-Nominal Operations .....	3-7
3.4.1	Analyze Trouble Ticket.....	3-7
3.4.2	Process L1 Product (Non-Nominal) .....	3-8
3.4.3	Recover From LPGS Failure.....	3-9

## Section 4. Global Libraries

4.1	Introduction.....	4-1
4.2	Design Overview.....	4-1
4.2.1	Library Overview.....	4-1
4.2.2	Design Considerations.....	4-1
4.3	Library Design .....	4-1
4.3.1	IAS Global Utilities .....	4-1
4.3.2	LPGS Global Utilities .....	4-2
4.3.3	IAS Global Database Utilities .....	4-4
4.3.4	LPGS Global Database Utilities .....	4-4

## Section 5. Process Control Subsystem

5.1	Introduction.....	5-1
5.2	Design Overview.....	5-1
5.2.1	Subsystem Software Overview.....	5-1
5.2.2	Design Considerations.....	5-3
5.2.3	Design Assumptions.....	5-3
5.3	Subsystem Design .....	5-3
5.3.1	System Initialization/Termination (PSI) Task.....	5-5

## REVIEW

5.3.2	Work Order Generator (PWG) Task.....	5-5
5.3.2.1	Task Overview .....	5-5
5.3.2.2	Initialization .....	5-5
5.3.2.3	Normal Operation.....	5-5
5.3.2.4	Error Handling.....	5-6
5.3.2.5	Design .....	5-6
5.3.3	Work Order Scheduler (PWS) Task .....	5-13
5.3.3.1	Task Overview .....	5-13
5.3.3.2	Initialization .....	5-13
5.3.3.3	Normal Operation.....	5-13
5.3.3.4	Error Handling.....	5-14
5.3.3.5	Design .....	5-14
5.3.4	Work Order Controller (PWC) Task .....	5-30
5.3.4.1	Task Overview .....	5-30
5.3.4.2	Initialization .....	5-30
5.3.4.3	Normal Operation.....	5-31
5.3.4.4	Error Handling.....	5-31
5.3.4.5	Design .....	5-31

## Section 6. Data Management Subsystem

6.1	Introduction.....	6-1
6.2	Design Overview.....	6-1
6.2.1	Subsystem Software Overview.....	6-1
6.2.2	Design Considerations/Assumptions .....	6-3
6.3	Subsystem Design .....	6-4

## REVIEW

6.3.1	DMS IF With ECS (DIE) Task .....	6-6
6.3.1.1	Task Overview .....	6-6
6.3.1.2	Initialization .....	6-6
6.3.1.3	Normal Operation .....	6-6
6.3.1.4	Error Handling.....	6-7
6.3.1.5	Design .....	6-7
6.3.2	DMS Ingest L0R Product (DIL) Task .....	6-16
6.3.2.1	Task Overview .....	6-16
6.3.2.2	Initialization .....	6-16
6.3.2.3	Normal Operation .....	6-16
6.3.2.4	Error Handling.....	6-17
6.3.2.5	Design .....	6-17
6.3.3	DMS Process L0R Product (DPL) Task .....	6-37
6.3.3.1	Task Overview .....	6-38
6.3.3.2	Initialization .....	6-38
6.3.3.3	Normal Operation .....	6-38
6.3.3.4	Error Handling.....	6-38
6.3.3.5	Design .....	6-38
6.3.4	DMS Format L1 Product (DFL) Task .....	6-42
6.3.4.1	Task Overview .....	6-42
6.3.4.2	Initialization .....	6-43
6.3.4.3	Normal Operation .....	6-43
6.3.4.4	Error Handling.....	6-43
6.3.4.5	Design .....	6-43

## REVIEW

6.3.5	DMS Xmit L1 Product (DXL) Task.....	6-57
6.3.5.1	Task Overview .....	6-57
6.3.5.2	Initialization .....	6-57
6.3.5.3	Normal Operation.....	6-57
6.3.5.4	Error Handling.....	6-58
6.3.5.5	Design .....	6-58
6.3.6	DMS Resource Manager (DRM) Task .....	6-65
6.3.6.1	Task Overview .....	6-65
6.3.6.2	Initialization .....	6-66
6.3.6.3	Normal Operation.....	6-66
6.3.6.4	Error Handling.....	6-66
6.3.6.5	Design .....	6-66
6.3.7	DMS Generate Reports (DGR) Task .....	6-77
6.3.7.1	Task Overview .....	6-77
6.3.7.2	Initialization .....	6-78
6.3.7.3	Normal Operation.....	6-78
6.3.7.4	Error Handling.....	6-78
6.3.7.5	Design .....	6-78
6.3.8	Subsystem Utilities .....	6-85

## Section 7. Radiometric Processing Subsystem

7.1	Introduction.....	7-1
7.2	Design Overview.....	7-1
7.2.1	Subsystem Software Overview.....	7-1
7.2.2	Design Considerations.....	7-2

## REVIEW

7.2.3	Design Assumptions.....	7-2
-------	-------------------------	-----

### Section 8. Geometric Processing Subsystem

8.1	Introduction.....	8-1
-----	-------------------	-----

### Section 9. Quality Assessment Subsystem

9.1	Introduction.....	9-1
9.2	Design Overview.....	9-1
9.2.1	Subsystem Software Overview.....	9-1
9.2.2	Design Considerations.....	9-1
9.2.2.1	Design Assumptions .....	9-1
9.2.2.2	Open Issues .....	9-3
9.2.3	Subsystem Error Handling .....	9-3
9.3	Subsystem Design .....	9-3
9.3.1	L1R Quality Assessment Task.....	9-5
9.3.1.1	Task Overview .....	9-5
9.3.1.2	Initialization .....	9-5
9.3.1.3	Normal Operation.....	9-5
9.3.1.4	Design .....	9-5
9.3.2	L1G Quality Assessment Task .....	9-8
9.3.2.1	Task Overview .....	9-8
9.3.2.2	Initialization .....	9-8
9.3.2.3	Normal Operation.....	9-9
9.3.2.4	Design .....	9-9
9.3.3	Quality Assessment User Interface Task .....	9-12

## REVIEW

9.3.3.1	Task Overview .....	9-12
9.3.3.2	Initialization .....	9-12
9.3.3.3	Normal Operation .....	9-12
9.3.3.4	Design Input, Output, Algorithm .....	9-12

## Section 10. Anomaly Analysis Subsystem

10.1	Introduction.....	10-1
10.2	Design Overview.....	10-1
10.2.1	Design Methodology.....	10-1
10.2.2	Solution Strategy .....	10-1
10.2.3	Design Considerations.....	10-1
10.2.3.1	Assumptions .....	10-1
10.2.3.2	Open Issues .....	10-2
10.2.4	Software Reuse Strategy .....	10-2
10.3	Subsystem Design .....	10-3
10.3.1	Functional Overview.....	10-3
10.3.2	Track Anomalies.....	10-3
10.3.2.1	Receive LPGS Anomaly .....	10-3
10.3.2.2	Receive Trouble Ticket.....	10-3
10.3.2.3	Approve L1 Product .....	10-7
10.3.2.4	Close Anomaly .....	10-7
10.3.2.5	Fail Product .....	10-7
10.3.3	Perform Anomaly Runs.....	10-7
10.3.3.1	Generate Diagnostic Work Order.....	10-7
10.3.3.2	Activate Diagnostic Work Order.....	10-9

## REVIEW

10.3.3.3	Resume Diagnostic Work Order .....	10-9
10.3.3.4	Generate Reprocessing Work Order .....	10-9
10.3.3.5	Generate Benchmark Work Order .....	10-9
10.3.3.6	Generate Product Request .....	10-9
10.3.4	View LPGS Data .....	10-9
10.3.4.1	View Data Files .....	10-11
10.3.4.2	View Database Stores .....	10-11
10.3.4.3	View Image .....	10-11
10.3.5	Generate History Report .....	10-11
10.3.6	Transmit Anomalies .....	10-11

## Section 11. Database Design

11.1	Introduction .....	11-1
11.2	Logical Design .....	11-1
11.2.1	Entities .....	11-1
11.2.2	Attributes .....	11-1
11.2.3	Relations .....	11-3
11.3	Physical Design .....	11-3
11.3.1	Tables .....	11-3
11.3.2	Indexes .....	11-4
11.3.3	Integrity Constraints .....	11-4
11.3.4	Database Storage Requirements .....	11-5
11.4	Database Sizing .....	11-6
11.5	Database Administration .....	11-6
11.5.1	Security .....	11-6

## REVIEW

11.5.2	Backup.....	11-11
11.5.3	Recovery.....	11-11

## Section 12. User Interface

12.1	Introduction.....	12-1
12.1.1	Design Considerations.....	12-1
12.1.2	Design Assumptions and Open Issues .....	12-1
12.1.3	Design/Development Tools .....	12-1
12.1.3.1	Common Tools .....	12-2
12.1.3.2	Analyst Tools .....	12-2
12.2	User Interface Functions.....	12-2
12.2.1	System Functions .....	12-2
12.2.1.1	Recovery .....	12-2
12.2.1.2	Manage Resources .....	12-2
12.2.1.3	Configure LPGS .....	12-3
12.2.1.4	Start Tasks.....	12-3
12.2.1.5	Restart Tasks .....	12-3
12.2.1.6	Shut Down Tasks.....	12-3
12.2.1.7	Exit.....	12-3
12.2.2	Monitor Functions.....	12-3
12.2.2.1	Database .....	12-3
12.2.2.2	View Event Log.....	12-3
12.2.2.3	Processing Statistics Report .....	12-3
12.2.3	Product Request Functions .....	12-4
12.2.3.1	Promote Product Request.....	12-4

## REVIEW

12.2.3.2	L0R Receipt.....	12-4
12.2.3.3	Product Request Information .....	12-4
12.2.3.4	Cancel Product Request .....	12-4
12.2.3.5	Work Order .....	12-4
12.2.4	Anomaly Analysis Functions.....	12-4
12.2.4.1	View .....	12-4
12.2.4.2	Work Order .....	12-5
12.2.4.3	Anomaly.....	12-5
12.2.4.4	Trouble Ticket.....	12-5
12.2.5	Quality Assessment Functions.....	12-5
12.2.5.1	View.....	12-6
12.2.5.2	Approve/Reject.....	12-6

## Figures

2-1	LPGS Symbols for Modules.....	2-3
2-2	LPGS Symbols for Connectors, Iterations, and Transactions .....	2-4
2-3	LPGS Symbols for Invocations and Couples .....	2-5
2-4	LPGS Hardware Architecture.....	2-7
2-5	LPGS Context Diagram.....	2-10
2-6	LPGS Level 0 DFD .....	2-11
5-1	PCS Context Diagram .....	5-2
5-2	PCS Level 0 DFD .....	5-4
5-3	PWG Structure Chart .....	5-7
5-4	PWS Structure Chart .....	5-15
5-5	PCW Structure Chart.....	5-32

## REVIEW

6-1	DMS Context Diagram.....	6-2
6-2	DMS Level 0 DFD.....	6-5
6-3	DIE Structure Chart.....	6-8
6-4	DIL Structure Chart.....	6-18
6-5	DPL Structure Chart.....	6-39
6-6	DFL Structure Chart.....	6-44
6-7	DXL Structure Chart.....	6-59
6-8	DRM Structure Chart.....	6-67
6-9	DGR Structure Chart.....	6-79
7-1	RPS Context Diagram.....	7-1
8-1	GPS Context Diagram.....	8-1
9-1	QAS Context Diagram.....	9-2
9-2	QAS Level 0 DFD.....	9-4
9-3	Q1R Structure Chart.....	9-6
9-4	Q1G Structure Chart.....	9-10
10-1	AAS Context Diagram.....	10-4
10-2	AAS Level 0 DFD.....	10-5
10-3	AAS DFD 1.0—Track Anomalies.....	10-6
10-4	AAS DFD 2.0—Perform Anomaly Runs.....	10-8
10-5	AAS DFD 3.0—View LPGS Data.....	10-10
11-1	LPGS ERD.....	11-2
11-2	LPGS Schema Diagram.....	11-4

## Tables

11-1	Entities and Their Descriptions.....	11-2
------	--------------------------------------	------

**REVIEW**

11-2 LPGS Database Sizing Assumptions ..... 11-7

11-3 Table Size and Row Estimates..... 11-8

11-4 Estimated Database Size Summary..... 11-10

**Appendix A. LPGS Requirements Traceability Matrix**

**Appendix B. LPGS Software Size Estimates**

**Appendix C. Database Table Definition Report**

**Abbreviations and Acronyms**

# Section 1. Introduction

---

## 1.1 Purpose

This document establishes the detailed software design for the Earth Science Data and Information System (ESDIS) Level 1 Product Generation System (LPGS). This design will be used to support the implementation of the LPGS.

## 1.2 Scope

This document presents the detailed design for the LPGS software configuration components. The scope includes the design of the LPGS globals, subsystems, database, and user interface.

The LPGS detailed design is based on an analysis of the requirements contained in the LPGS Functional and Performance Requirements Specification (F&PRS), the LPGS Operations Concept, and the LPGS System Design Specification (SDS). The design also is based on the interface control documents (ICDs) between the LPGS and the Earth Observing System (EOS) Data and Information System (EOSDIS) Core System (ECS) and between the LPGS and the Image Assessment System (IAS).

## 1.3 Applicable Documents

The documents listed in the subsections that follow contain details about the LPGS and external systems.

### 1.3.1 Requirements Documents

The following documents were used in developing the LPGS system design:

1. National Aeronautics and Space Administration (NASA)/Goddard Space Flight Center (GSFC), 510-FPD/0196, *Level 1 Product Generation System (LPGS) Functional and Performance Requirements Specification*, February 1997
2. —, 510-3OCD/0296 (CSC 10034093), *Level 1 Product Generation System (LPGS) Operations Concept*, February 1997
3. —, *Earth Science Data and Information System (ESDIS) Mission-Specific Requirements for the Landsat 7 Mission L1 Processing*, Draft, January 1997
4. —, *Landsat 7 Level 1 Product Generation System (LPGS) Project Management Plan*, Draft, May 1996
5. —, *Image Assessment System (IAS) Design Specification*, December 1996
6. —, 514-4ICD/0197 (CSC 10037996), *Interface Control Document Between the Image Assessment System (IAS) and the Level 1 Product Generation System (LPGS)*, Review, August 1997

## REVIEW

7. —, 209-CD-029-001, *Interface Control Document Between the EOSDIS Core System (ECS) and the Level 1 Product Generation System*, Draft, June 1997
8. —, 510-4SDS/0196 (CSC 10034686), *Earth Science Data and Information System (ESDIS) Level 1 Product Generation System (LPGS) System Design Specification*, March 1997
9. —, *Image Assessment System (IAS) Critical Design Specification*, April 1997
10. —, 430-11-06-007-0, *Landsat 7 System Zero-R Distribution Product Data Format Control Book, Volume 5, Book 1*, May 1997
11. —, 510-3DFC/0197, *ESDIS Level 1 Product Generation System (LPGS) Output Files Data Format Control Book*, May 1997
12. —, 510-1IDD/0197 (CSC 10037648), *Earth Science Data and Information System (ESDIS) Level 1 Product Generation System (LPGS) Interface Definitions Document (IDD)*, Review, August 1997
13. —, 510-3SUG/0297 (CSC 10037610), *Earth Science Data and Information System (ESDIS) Level 1 Product Generation System (LPGS) User's Guide*, Draft, August 1997
14. EROS Data Center, *Landsat 7 IAS Geometric Processing Subsystem Detailed Design Specification*, March 1997

### 1.3.2 Reference Documents

The following documents contain additional details about the LPGS, ESDIS and Landsat 7 systems and projects, and external systems:

1. NASA/GSFC, 560-3OCD/0194, *Landsat 7 Processing System (LPS) Operations Concept, Revision 2*, April 1996
2. —, *Landsat 7 Detailed Mission Requirements*, May 1995
3. —, *Image Assessment System (IAS) Operations Concept*, December 1994
4. —, 430-15-01-001-0, *Landsat 7 Image Assessment System (IAS) Element Specification*, Baseline, October 1996
5. —, 514-1ICD/0195, *Interface Control Document Between IAS and LPS*, January 1996
6. —, 209-CD-013-004, *Interface Control Document Between ECS and the Landsat 7 System*, August 1996
7. —, *Interface Requirements Document (IRD) Between ECS and the Landsat 7 System*
8. —, 505-41-18, *IRD Between EOSDIS and MITI ASTER GDS Project*, July 1995
9. —, 505-41-13, *IRD Between EOSDIS and the Landsat 7 System*, July 1995
10. EOSDIS Core System Project, 223-CD-001-002, *ECS External Data Traffic Requirements*, August 1996

## REVIEW

11. —, 604-CD-002-003, *ECS Operations Concept for the ECS Project: Part 2B - ECS Release B*, March 1996
12. —, 305-CD-027-002, *Release-B SDPS Data Processing Subsystem Design Specification*, March 1996
13. —, 604-CD-003-002, *ECS Operations Concept for the ECS Project Part 2A - ECS Release A*, November 1995
14. —, 305-CD-029-002, *Release B CSMS System Management Subsystem Design Specification for the ECS Project*, July 1994
15. —, 194-207-SE1-001, *System Design Specification for the ECS Project*, June 1994
16. —, 305-CD-024-002, *Release B SDPS Data Server Subsystem Design Specification for the ECS Project*
17. —, *Landsat 7 System and Operations Concept*, October 1994
18. —, *Mission Operations Concept for the Landsat 7 Ground System*, Draft, June 1995
19. United States Geological Survey (USGS)/National Oceanic and Atmospheric Administration (NOAA), *Index to Landsat Worldwide Reference System (WRS) Landsats 1, 2, 3, and 4*, 1982
20. Computer Sciences Corporation, *SEAS System Development Methodology, Version 3*, July 1996

### 1.4 Definitions

The following terms, as defined in this section, are commonly used throughout this document:

- *Interval*—The time duration between the start and stop of an imaging operation (observation) of the Landsat 7 Enhanced Thematic Mapper Plus (ETM+) instrument
- *Level 0R (LOR) digital image*—Spatially reformatted, demultiplexed, and unrectified subinterval data
- *Level 0R (LOR) product*—LOR digital image plus radiometric calibration, attitude, and ephemeris data in Hierarchical Data Format (HDF) EOS unencapsulated format. The LOR product consists of the following files:
  - LOR digital image
  - Internal calibrator (IC) data—Calibration data file containing all the calibration data received on a major frame basis for a given subinterval
  - Mirror scan correction data (MSCD)—Scan direction and error information for a given subinterval
  - Payload correction data (PCD)—Information on spacecraft attitude and ephemeris, including quality indicators for each subinterval

## REVIEW

- Metadata—Descriptive information about Level 0 (L0) digital image, names of appended files associated with the image, and quality and accounting information
- Calibration parameter file (CPF)—Formatted file containing gains, biases, and offsets for the instrument and detectors
- *Level 1R (L1R) digital image*—Radiometrically corrected, but not geometrically resampled
- *Level 1R (L1R) product*—L1 product packaged by the LPGS and distributed by the ECS to the customer consisting of the following in HDF format:
  - L1R digital image
  - IC data—Calibration data file containing all the calibration data received on a major frame basis for a given subinterval
  - Consensus MSCD—Scan direction and error information for a given subinterval
  - Consensus PCD—Information on spacecraft attitude and ephemeris, including quality indicators for each subinterval
  - Metadata—Descriptive information about the L1 digital image, names of appended files associated with the image and accounting information
  - CPF—Formatted file containing gains, biases, and offsets for the instrument and detectors
  - Geolocation table—Scene corner coordinates and scan line numbers
- *Level 1G (L1G) digital image*—Radiometrically corrected and resampled for geometric correction and registration to geographic map projections
- *Level 1G (L1G) product*—L1 product packaged by the LPGS and distributed by the ECS to the customer; includes, for all requested bands, Fast Argonne System for Transport (FAST-C) or Geographic Tag(ged) Image File Format (GeoTIFF) format L1G image and associated data accommodated by the format; HDF format L1G image; and metadata
- *Procedure*—A template used to create work orders. It includes an ordered list of work order scripts and the default parameters associated with each script
- *Product generation request*—Request received from the ECS directing the LPGS to generate a specific L1R or L1G product. The product generation request includes
  - L0R subinterval identifier
  - Start and stop scan lines
  - Output product (L1R or L1G)
  - Output format

## REVIEW

- Band selection
- Map projection option and project parameters
- Grid cell size
- Resampling filter
- Orientation
- Calibration method
- *Subinterval*—Segment of time corresponding to a portion of an observation within a single Landsat 7 contact period
- *User Request File (URF)*—See product generation request
- *Work order*—Detailed instructions for the Level 1 (L1) processing of a specific product generation request. A work order is constructed from an ordered list of work order scripts. It also encompasses the parameters associated with each script
- *Work order script*—A UNIX script used to run one or more programs as a part of a work order
- *Worldwide Reference System (WRS) scene*—Digital image that covers an area equivalent to one of the 57,784 scene-centers (233 paths x 248 rows areas) defined by the WRS structure

### 1.5 Document Organization

This document is organized into 12 sections and 3 appendixes. Section 1 describes the purpose and scope of the document, identifies the applicable documents, and defines some of the LPGS terminology. Section 2 provides an overview of the detailed design including the design methodology; an overview of the LPGS in terms of hardware, software, and external interfaces; and the design considerations. Section 3 provides descriptions of operational activities associated with LPGS functions. Section 4 describes the major sections of the global library. Sections 5 through 10 present the detailed design of the LPGS subsystems. Section 11 describes the LPGS database. Section 12 provides descriptions of each of the user interface functions. Appendix A provides the LPGS requirements traceability matrix, which maps requirements to subsystems, hardware, and operations. Appendix B gives the LPGS software estimates, and Appendix C discusses the database table definition report. A list of abbreviations and acronyms used in the document follows the appendixes.

# Section 2. Detailed Design Overview

---

This section provides an overview of the LPGS detailed design. It discusses the design methodology, system overview, and design considerations.

## 2.1 Design Methodology

This section provides a brief description of the methods used to perform the detailed design of the LPGS, including the design process, products, and conventions.

### 2.1.1 Design Process

The LPGS detailed design was developed using the *SEAS System Development Methodology (SSDM)* (Reference Document 20) tailored to meet the requirements of the LPGS project environment. The LPGS design has been accomplished through the following major activities:

- Development of an LPGS top-level software architecture that is based on an SSDM structured design and conforms to the selected hardware configuration and constraints
- Design of the LPGS database, which is based on the refinement of a logical model of the LPGS data
- Design of a user interface (UI) for the LPGS, based on software requirements and operations concepts
- Design of the LPGS subsystems using a computer-aided software engineering (CASE) tool, Cadre/Teamwork, which supports the structured design methodology
- Analysis of other systems (most notably IAS) for potential software re-use

### 2.1.2 Design Products

Several products are produced as a result of the critical design phase of the LPGS. They include the following:

- A model of the LPGS design in the Cadre/Teamwork CASE tool that describes the software design. This model includes the following:
  - Context diagrams for each subsystem and decomposed data flow diagrams (DFDs)
  - Process specifications (P-Specs) that describe each process within a DFD
  - For each LPGS task, structure charts that are graphical representations of the hierarchy of the LPGS modules. Structure charts consist of modules and the data and/or control couples passed between modules
  - Module specifications (M-Specs) that describe a module's function and how the module performs the function

## REVIEW

- Program design language (PDL) for all top-level modules
- Data dictionary that provides definitions for data items in the LPGS software design
- This detailed design specification (DDS)

### 2.1.3 Design Conventions

The LPGS design is expressed in a notation adapted from E. Yourdon and L. Constantine and supported by the Cadre/Teamwork CASE tool. The notation is illustrated in Figures 2-1, 2-2, and 2-3. The notation is consistent with SSDM Standard 4205 but includes the following enhancements and exceptions:

- Unit representations on structure charts include only a title; they do not include a purpose statement. The conditional execution symbol is used for transaction centers as well.
- Signals (UNIX software interrupts) are represented by asynchronous invocation from an off-sheet connector without a source.
- Special notations are used for LPGS global, file access, and re-use routines, as illustrated in Figure 2-1.

## 2.2 LPGS General Overview

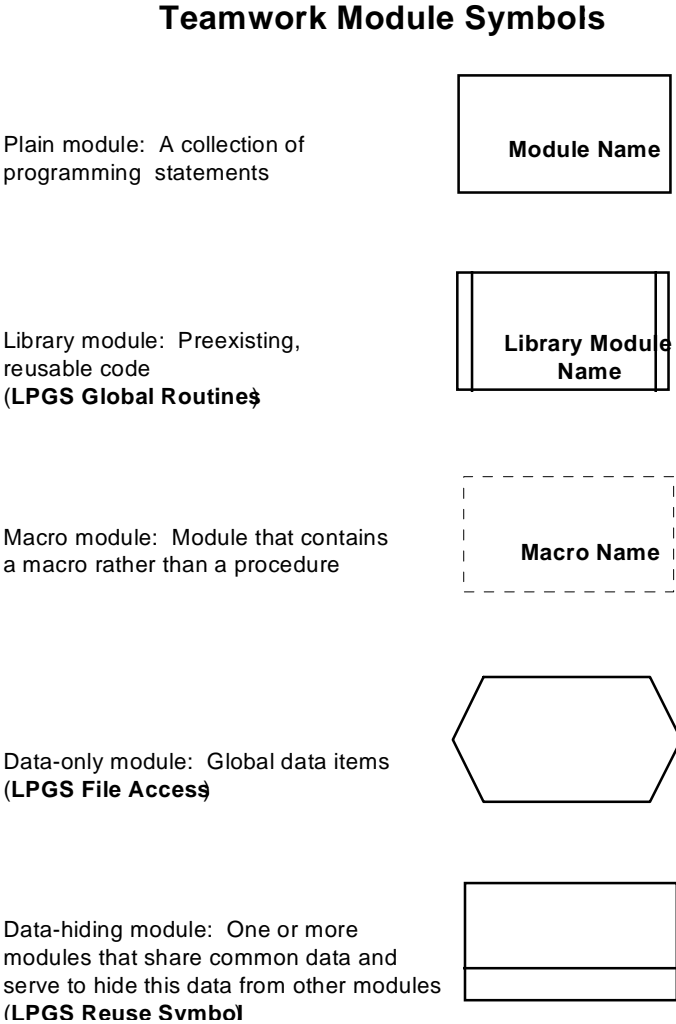
The LPGS receives L1 product generation requests and distributes generated products to customers through the ECS at the Earth Resources Observation System (EROS) Data Center (EDC) on a first-ordered, first-processed basis.

The LPGS is the responsibility of the ESDIS Project and is to be installed at the EDC Distributed Active Archive Center (DAAC) to provide product generation and distribution support for a Landsat 7 minimum mission life of 5 years.

The LPGS produces L1 data products in electronic format for partial ETM+ subintervals (.5 to 3 WRS scene equivalents) based on customer requests. The LPGS can produce a daily volume of 25 WRS scene equivalents of L1R, radiometrically corrected and L1G, digitally resampled for geometric correction and geographic registration. The LPGS can create digital images projected to different coordinate reference systems for any subset of the eight spectral channels collected by the ETM+ instrument or in different output formats according to other options specified in the customer's request.

ECS User Services provides the interface that customers use to request Landsat 7 L1 products. The ECS supplies the LPGS with the customer requests in the form of product generation requests. A product generation request identifies the subinterval, the type of product (L1R or L1G), and various options used to control L1 processing. The ECS places product generation requests in a known ECS disk location for "just in time" processing. LPGS periodically polls this location for new requests. When the LPGS finds a new request, it transfers the request to the LPGS disk and updates the LPGS database with the request information.

Figure 2-1. LPGS Symbols for Modules



**Figure 2-2. LPGS Symbols for Connectors, Iterations, and Transactions**

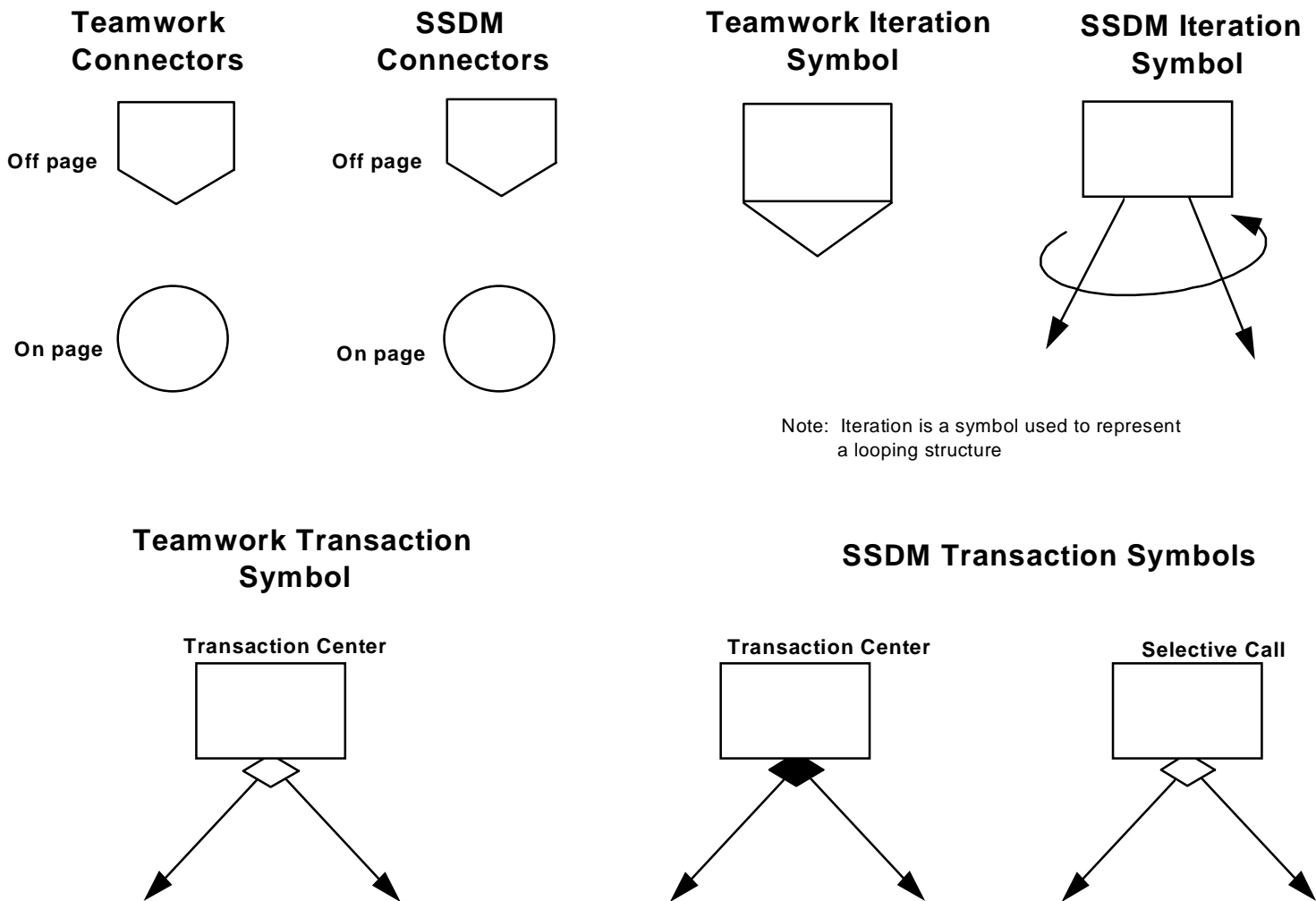
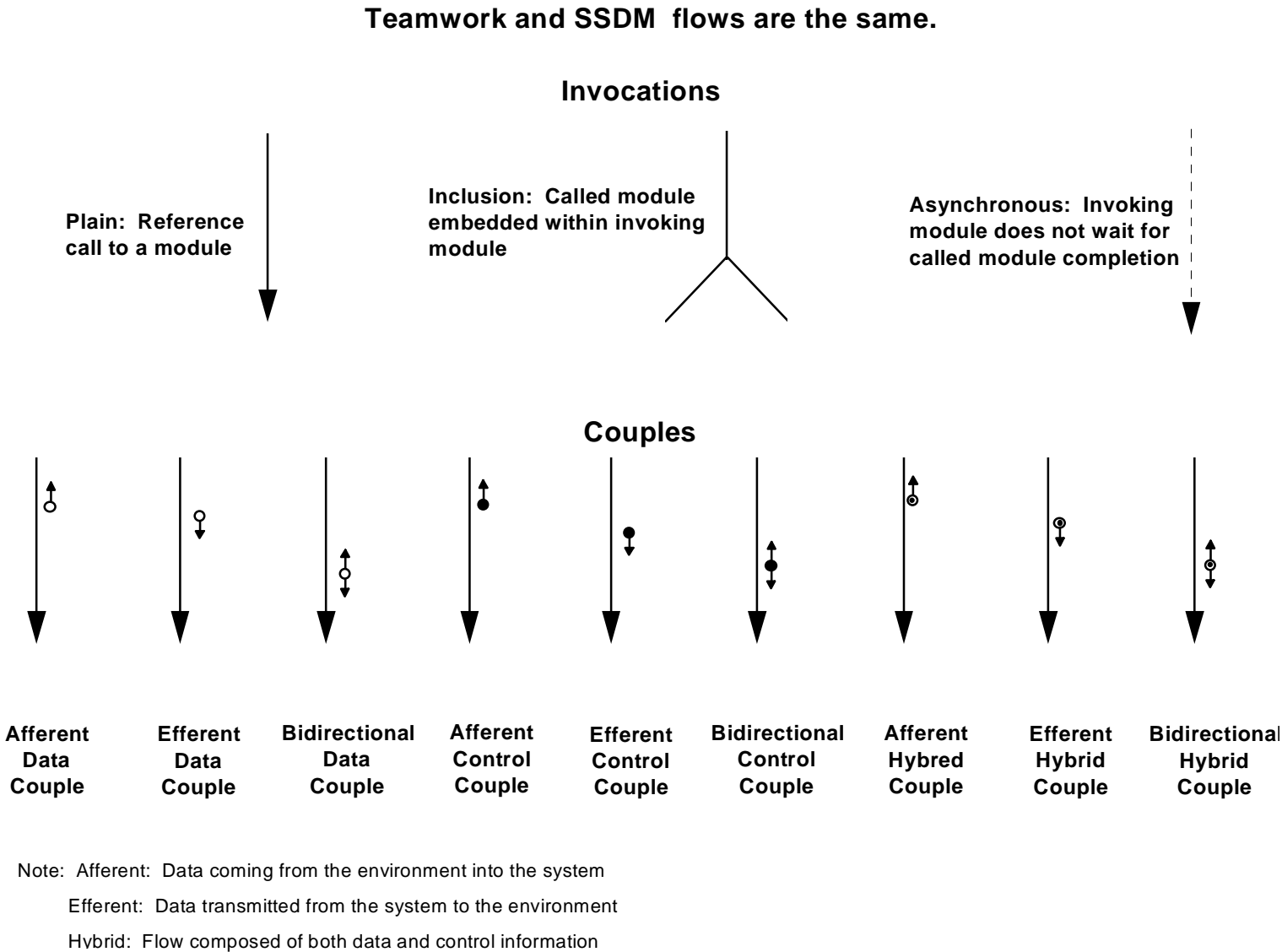


Figure 2-3. LPGS Symbols for Invocations and Couples



## REVIEW

Before the LPGS can begin L1 processing, it must ingest the LOR product needed for processing. The LPGS requests the LOR product from the ECS. The ECS subsets the LOR data, stores it on the ECS disk, and notifies the LPGS of the LOR product availability. The LPGS retrieves the LOR product from the ECS and verifies that the product is correct and complete.

After the LOR product needed for a product generation request has been ingested, the LPGS initiates L1 processing by creating an LPGS work order. A work order is an internal mechanism used to control and monitor the L1 processing. As a part of work order processing, the LPGS runs a number of UNIX scripts that perform the radiometric and geometric corrections, quality assessments, and product preparation. Each of the work order scripts has parameters associated with it. Many of the parameter values are determined by the customer-supplied options in the product generation request.

After the L1 product has been produced, the LPGS notifies the ECS that the product is ready for distribution. The LPGS creates an L1 product availability notice and places the notice in a known location on the LPGS disk. The ECS periodically polls this location for new product availability notices. When the ECS finds a new notice, it transfers the product to the ECS disk space, performs its ingest function, and delivers the product to the customer.

Trending data are created in the LPGS database as a part of the work order processing. Because the IAS processes fewer scenes than the LPGS, the IAS plans to use the LPGS trending data to augment the IAS trending data. The IAS periodically pulls the trending data from the LPGS database.

The LPGS interfaces with the ECS within the EDC DAAC and with the IAS. (Consult the appropriate system ICD for more information concerning a specific interface.)

### 2.2.1 Hardware

The LPGS hardware will be located within the EDC DAAC in Sioux Falls, South Dakota. Figure 2-4 presents the LPGS architecture.

#### 2.2.1.1 LPGS System Hardware

The LPGS operations support system performs L1 processing, provides storage for the production control database, and provides temporary storage for ingesting incoming LOR products and outgoing L1 products. A backup system can be used, after proper configuration, as a production system if the prime system fails. The operations support system consists of one Silicon Graphics, Inc. (SGI) Origin 2000 server, one set of redundant array of inexpensive devices (RAID) disk arrays, two SGI 02 workstations, one X terminal, one 8-millimeter (mm) tape drive, and a network printer. Additional hardware includes an Origin 2000 server, SGI 02 workstation, and an X terminal to be used for software maintenance and testing and also as a backup in case a failure occurs in the operations support system. Any of the three SGI 02 workstations and two X terminals can be configured to interface with either the operations support system or the backup system.

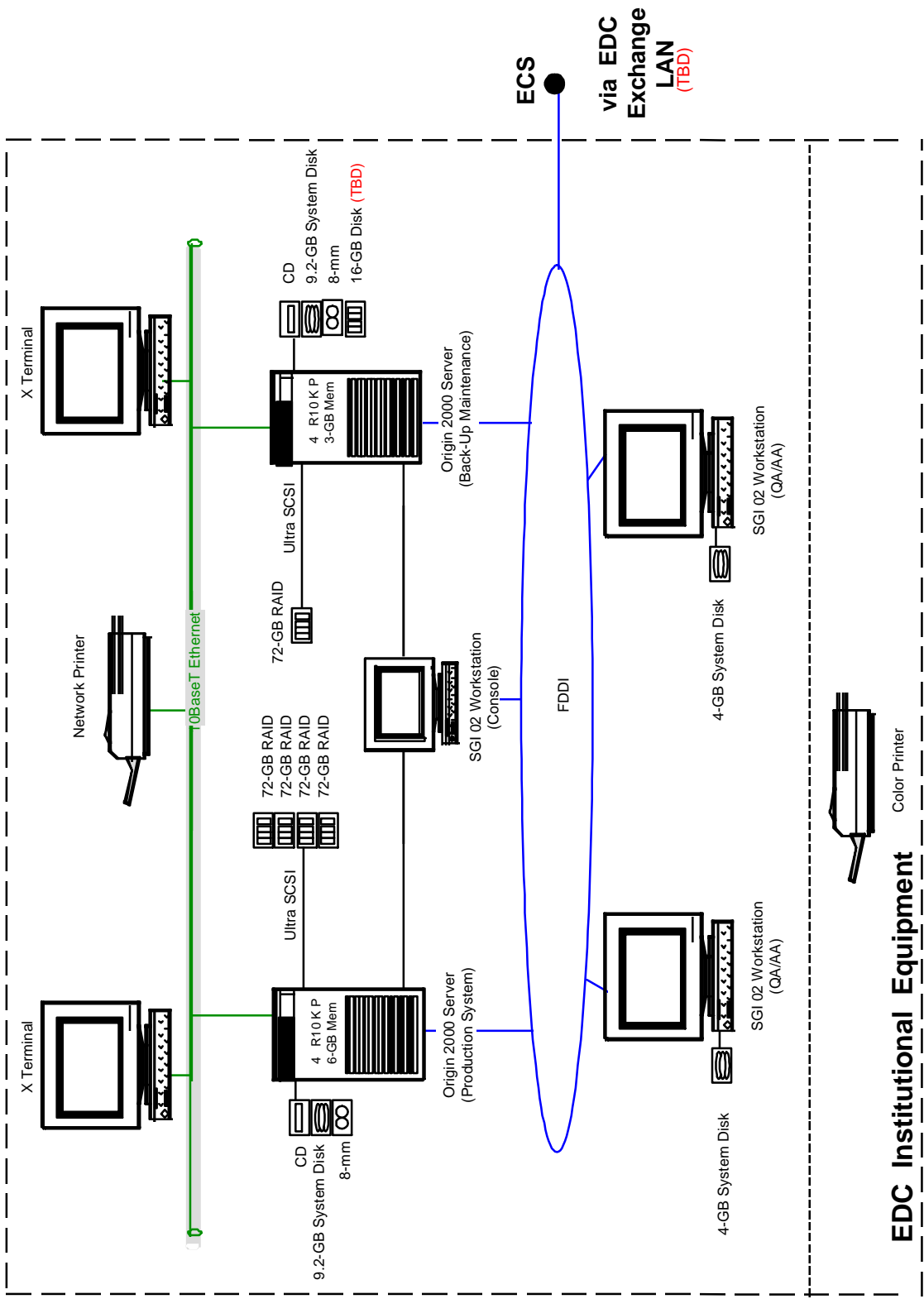


Figure 2-4. LPGA Hardware Architecture

## **REVIEW**

### **2.2.1.1.1 SGI Origin Server**

The SGI Origin servers are multiprocessor systems designed for distributed computing environments. Their parallel architecture is based on a 1.2-gigabyte (GB)-per-second (GBps) system bus and can support up to 256 GB of random access memory (RAM). It is a modular building block of processors, input/output (I/O), memory, system bandwidth, power supplies, and chassis. A single deskside system module supports 1 to 8 million instructions per second (MIPS) R10000s that can be rack mounted and expanded to 128 processors. The LPGS operations server initially will have four processors with 6 GB of RAM. Among the items included with the standard subsystem are an Ethernet controller, Versa Module European (VME/64) controller, multiple small computer system interface (SCSI) controllers, and parallel and serial ports. The operations server consists of a compact disc read-only memory (CD-ROM) drive, a 9.2-GB system disk, and an 8-mm tape drive. The second system, normally used for testing, will be identical to the operations server except it will have an additional 16 GB of system disk space for storing images and only 3 GB (instead of 6 GB) of RAM initially. There is a plan to increase the RAM for the backup system to 6 GB.

The LPGS requires a multiprocessor solution to satisfy its computational requirements, which are beyond the current capacity of a single processor.

### **2.2.1.1.2 SGI 02 Workstation Hardware**

The SGI 02 workstations are based on the MIPS R10000 processor. These workstations are used by the LPGS analyst to perform quality assessment (QA) and anomaly analysis (AA).

The QA workstation is connected to the Origin 2000 server on the console port, and the AA workstation is connected to the LPGS server via the fiber-optic data distribution interface (FDDI) local area network (LAN). The display resolution is 1280 x 1024. These workstations have a 4-GB disk and 64 MB of RAM.

### **2.2.1.1.3 X Terminals**

This hardware configuration consists of two X terminals. These terminals are used by the LPGS operators to monitor and control processing on the operations and test systems. The terminals are connected to the Origin 2000 server via the Ethernet LAN.

### **2.2.1.1.4 FDDI LAN**

The FDDI LAN connects the operations support and backup systems with the EDC DAAC. The LAN is rated at 100 megabits per second (Mbps).

### **2.2.1.1.5 SCSI Controllers**

The SCSI 2 controller is used to connect to the CD-ROM, system disk, and 8-mm tape drive. Multiple Ultra-SCSI controllers are used to connect to the RAID array to get better disk I/O performance.

## REVIEW

### 2.2.1.1.6 RAID Array

The Ciprico 6900 series of disk arrays supports the Ultra-SCSI interface at 40 megabytes per second (MBps). Each array has a capacity of up to 72 GB. Five disk arrays giving a total of 360 GB of storage were selected for storing images and associated data. Initially, four disk arrays will be configured for the operations system, and one disk array will be configured for the backup system.

### 2.2.2 Software

The LPGS software architecture incorporates custom LPGS applications, commercial off-the-shelf (COTS) products, and system and support applications to meet its functional and performance needs.

#### 2.2.2.1 LPGS Application Software

The LPGS context diagram (Figure 2-5) depicts the LPGS external interfaces. The LPGS Level 0 data flow diagram (DFD) (Figure 2-6) shows the interactions between the LPGS subsystems and their external interfaces.

The scope and the allocation of requirements for each subsystem were driven by the LPGS requirements. These requirements have been further divided between operations, hardware, and software and analyzed to form this DDS (see Appendix A).

The LPGS is composed of six major subsystems: the Data Management Subsystem (DMS), Process Control Subsystem (PCS), Radiometric Processing Subsystem (RPS), Geometric Processing Subsystem (GPS), Quality Assessment Subsystem (QAS), and Anomaly Analysis Subsystem (AAS). DMS maintains the primary external interfaces. AAS indirectly interfaces with the ECS Trouble Ticket System via REMEDY.

The following paragraphs describe the purpose and function of each of the major LPGS subsystems. Complete detailed designs for each subsystem are presented in subsequent sections.

- **Data Management Subsystem**—The DMS maintains and provides access to LPGS data stores. The DMS handles communication protocols with LPGS external interfaces and ingests and formats files for use by other LPGS subsystems, providing cursory quality checks where needed. The DMS provides formatting and packaging of L1 output and makes these data available to external systems. The DMS also maintains LPGS disk space, populating temporary storage with data from ingested files.
- **Process Control Subsystem**—The PCS controls LPGS production planning and processing. The PCS takes product generation requests and sets up, monitors the status of, and controls processing of LPGS work orders. The PCS manages and monitors LPGS resources and provides processing status in response to operator requests.
- **Radiometric Processing Subsystem**—The RPS converts the brightness of the LOR image pixels to absolute radiance in response to user requests and in preparation for geometric correction. The RPS performs radiometric characterization of LOR images by

REVIEW

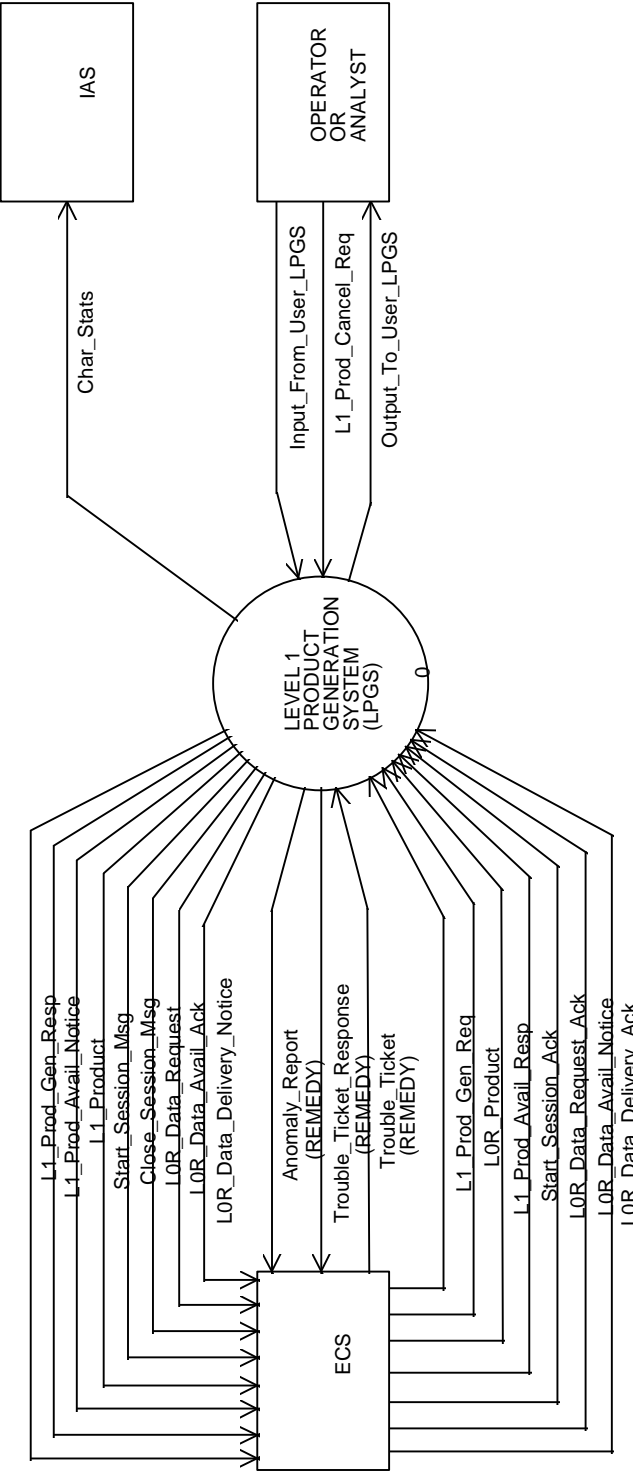
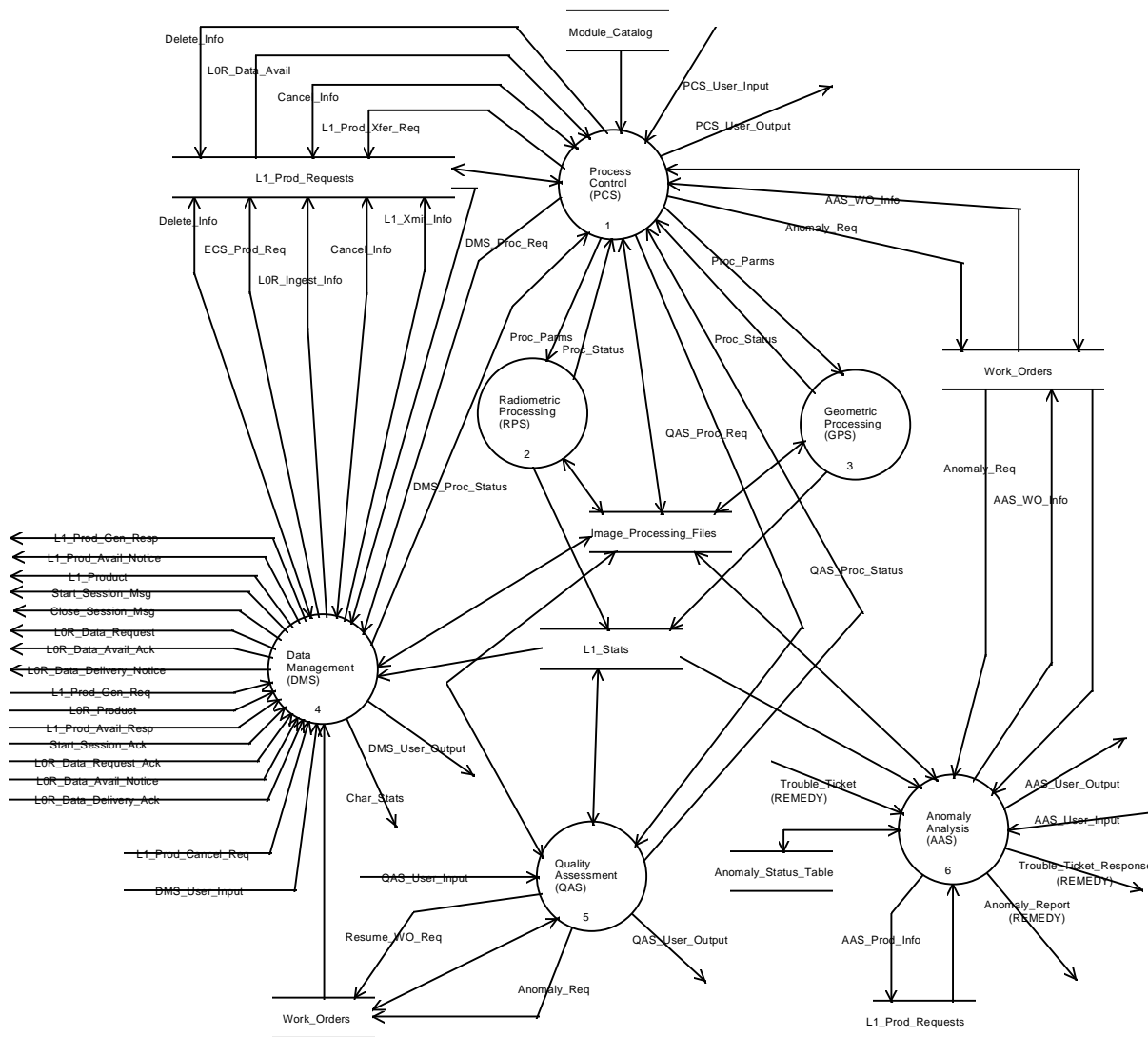


Figure 2-5. LPGS Context Diagram

## REVIEW



**Figure 2-6. LPGS Level 0 DFD**

locating radiometric artifacts in images. The RPS provides the results of characterizations performed and the processing status for use by external elements and other LPGS subsystems. The RPS uses applicable algorithms to correct for the radiometric artifacts found, then converts the image to absolute radiance using IC data.

- **Geometric Processing Subsystem**—The GPS creates systematically corrected L1G imagery from L1R products. The GPS provides the results of characterizations performed and the processing status for use by external elements and other LPGS subsystems. The GPS prepares a resampling grid, re-creates the L1R image within the grid, and applies one of three optional resampling techniques. The GPS performs

## REVIEW

sophisticated satellite geometric correction to create the image according to the user-specified map projection and orientation.

- **Quality Assessment Subsystem**—The QAS generates and assembles postproduction information about image artifacts and effects that were not corrected, and it produces a summary of the processed image quality. The QAS performs automated QA after radiometric and geometric correction of images has been made. The QAS provides tools for visual inspection of images after radiometric correction, geometric correction, and final product formatting.
- **Anomaly Analysis Subsystem**—The AAS analyzes L1 images and associated postproduction information about image artifacts and effects to resolve image production anomalies. AAS investigates problems encountered during LPGS production processing or problems reported by the end user after receipt of the L1 product. The AAS provides results of such problem analysis to the ECS for further investigation.

### 2.2.2.2 LPGS Support Software

The LPGS support software consists of COTS system and application software components that either satisfy LPGS requirements or are needed to perform specific functions of the LPGS. The support software includes the following:

- IRIX operating system
- Oracle database management system (DBMS)
- Oracle Structured Query Language (SQL) Forms and X/Motif Graphic User Interface (GUI) Tools
- National Center for Supercomputer Applications (NCSA) HDF
- EOS-HDF toolkit
- Imaging viewing product (TBD)
- Object Descriptive Language (ODL)/PVL manipulation routines
- EgFtp
- GeoTIFF and FAST-C read/write routines
- REMEDY

### 2.2.3 External Interfaces

As shown in Figure 2-5, the LPGS has external interfaces with the ECS and the IAS.

## **REVIEW**

### **2.2.3.1 Interface With the ECS**

The primary LPGS interface is with the ECS within the EDC DAAC. The LPGS retrieves L1 product generation requests from the ECS. L1 product generation requests are processed by the LPGS, which submits an L0R data request to the ECS. The LPGS then electronically retrieves L0R products, including CPFs, in response to the request. After L1 processing is complete, the images and other pieces of the L1 product are provided electronically to the ECS. The ECS supplies information regarding L1 product problems reported by the customer through a trouble ticket system using REMEDY. The LPGS responds to trouble tickets through REMEDY. The LPGS also reports anomalies detected during L1 processing that could not be resolved by the LPGS through the ECS trouble ticket system.

### **2.2.3.2 Interface With the IAS**

During L1 image processing, various characterizations for effects from saturated detectors, coherent noise, and memory effects are performed. The LPGS collects the results of the characterizations, and the IAS periodically retrieves them from the LPGS database.

## **2.3 Design Considerations**

This subsection presents the concepts and philosophy that drive the LPGS design.

### **2.3.1 Global Library Areas**

A library exists that contains global routines to provide general purpose processing throughout the LPGS. In addition, the LPGS uses global routines to ensure that the radiometric and geometric processes operate in the same manner on both the LPGS and the IAS.

### **2.3.2 Database Interface**

A trade-off study (performed by the IAS Project) resulted in the decision to place all database access through embedded SQL in separate database access routines (DBARs) that contain little more than the SQL statements themselves. The rationale for doing so is that these smaller units run less risk of encountering compile-time problems with the Oracle embedded SQL preprocessor.

The product used to implement the database interface is the Oracle tool Pro\*C. Pro\*C is a programming tool that allows SQL statements to be embedded in C source programs. It provides a convenient and easy interface that allows C applications to access Oracle directly.

### **2.3.3 User Interface**

Many of the LPGS functions are automated, but the LPGS also provides manual controls and other interactive functions. The LPGS UI is the primary interface between the operator/analyst and the LPGS subsystems. The LPGS UI will be implemented using Oracle Forms. See Section 12 for additional information on the UI.

## **REVIEW**

### **2.3.4 Interprocess Communication**

The LPGS provides interprocess communication (IPC) between LPGS tasks and between the UI and LPGS tasks. The LPGS design currently supports three types of IPC.

#### **2.3.4.1 Database Communication**

The primary method for IPC is by updates to various fields in the database. For instance, the state field in the product\_requests table indicates when a request needs LOR data, when it is ready for work order processing, and when it is ready to be shipped to the ECS. In general, the LPGS tasks periodically poll the database for work based on the values of one or more fields.

#### **2.3.4.2 Directives**

The LPGS provides a second method of IPC through the use of directives. The IPC directive method is re-used from the IAS design. It provides communication from the UI to the LPGS tasks. The UI inserts the directives into the IPC database table. The LPGS tasks periodically check the IPC table for new directives.

#### **2.3.4.3 Oracle Pipes**

The third method of IPC is through the use of Oracle pipes. This method is also used to communicate between the UI and the LPGS tasks. The UI writes to the appropriate pipe, and the receiving task reads from it.

### **2.3.5 Error Handling Philosophy**

The LPGS error handling philosophy provides for error detection by a monitoring and reporting process. Generally, each process that performs a function is invoked and monitored by a controlling process. The LPGS error handling philosophy distinguishes a number of classes of errors. A primary distinction is made between correctable and uncorrectable errors.

#### **2.3.5.1 Error Reporting**

LPGS error reporting consists of detailed error and status messages delivered throughout processing. Messages are reported to the event log, system log, and/or work order log, depending on the LPGS task. The event log is stored in the LPGS database. The system and work order logs are text files written to disk. The LPGS reports each detected error by the unit that first encounters the error.

Error and status messages requiring the user's (operator or analyst) attention are written to the LPGS event log. The user can query these events based on work order ID, work order script, program name, time, message severity, or product request ID.

Error and status messages requiring the LPGS software developer's attention are written to the LPGS system log. To aid in troubleshooting, the time, filename, process ID, and source code line number are appended to each message.

## REVIEW

Error and status messages encountered by radiometric and geometric processing software are written to the work order log. To aid in troubleshooting, the time, filename, process ID, and source code line number are appended to each message.

### **2.3.5.2 Error Processing**

The LPGS design treats most errors as uncorrectable and terminates execution gracefully when an error occurs. This decision is intended to minimize development costs without impacting system requirements. The decision is feasible because operations on the LPGS are repeatable.

#### **2.3.5.2.1 Correctable Errors**

In the LPGS, a correctable error is reported and handled in the unit that detects it.

#### **2.3.5.2.2 Uncorrectable Errors**

An uncorrectable error in the functioning process is detected by the monitoring process. In most cases, correction after a fatal error is unnecessary; the LPGS's state remains unchanged until the process terminates successfully.

Uncorrectable errors fall into two categories, fatal and nonfatal:

- Fatal errors—These errors cause a process to terminate abnormally and immediately. No opportunity for recovery exists when the error occurs. Catastrophic hardware failures or receipt of an operating system signal such as UNIX SIGKILL are events that can cause a process to terminate abnormally.
- Nonfatal errors—These errors make further processing impossible but do not abort the process. The process is able to respond to the error at the time of its occurrence.

### **2.3.6 File Maintenance**

The LPGS design produces the following types of files:

- Work order log
- L1R and L1G intermediate output files
- Formatted L1R and L1G product files
- ECS interface protocol files
- Temporary files created by various LPGS software components and deleted at program termination or the analyst's completion of assessment activities

The ECS protocol files are deleted by the appropriate DMS task when they are no longer required. The other types of files are marked for deletion after the product has been transferred successfully to the ECS or after AAS has determined that the LPGS is unable to produce the product. The LPGS periodically performs disk clean up by deleting files marked for deletion.

## REVIEW

### Section 3. LPGS Operational Scenarios

---

#### 3.1 Introduction

Operational scenarios describe the allocation and flow of LPGS operations activities between the operations staff and subsystems. The allocation of LPGS requirements to subsystems, hardware, and operations is found in Appendix A. Although the majority of nominal LPGS functions—such as product ingest, formatting and distribution, radiometric and geometric correction, and initial quality assessments—are intended to be performed automatically, without operator intervention, operations staff have roles in both nominal and non-nominal activities. The scenarios are meant to provide the basis for dialogue between systems engineering, development, and operations representatives. Key operations activities have been categorized as nominal and non-nominal as indicated below.

Nominal Activities	Non-Nominal Activities
<ol style="list-style-type: none"><li>1. Start up LPGS</li><li>2. Shut down LPGS</li><li>3. Process L1 product (nominal end-to-end L1 processing flow)</li><li>4. Cancel L1 processing</li><li>5. Retrieval of characterization results by IAS</li></ol>	<ol style="list-style-type: none"><li>1. Analyze trouble ticket</li><li>2. Process L1 product (non-nominal)</li><li>3. Recover from LPGS failure</li></ol>

The flow of operations activities performed by operations staff and LPGS subsystems to accomplish key LPGS functions are detailed in the sections that follow.

#### 3.2 Operations Staff

Operations staff are required to perform activities that the LPGS subsystems cannot perform automatically. Operations activities performed by the operations staff can be grouped into system operations, production control, QA, and AA categories. System operations, performed by the system operator (Sys Opr), include initiating system startup and shutdown, configuring system software and hardware, backing up system software in support of both nominal and contingency operations, and monitoring system status. Production control operations, performed by the production operator (Prod Opr), include monitoring the L1 processing activities, manually modifying LPGS work orders, initiating statistics generation, and canceling product generation requests. QA and AA activities are performed by the analyst. The analyst visually inspects images before distribution, analyzes and inspects processing quality information, and resolves processing anomalies found both before and after distribution of the L1 product. A single operations staff member can support multiple operations staff positions, and multiple staff members can support a single operations position as hardware architecture permits.

## REVIEW

### 3.3 Nominal Operations

Nominal operations scenarios are presented in the sections below according to the sequentially numbered steps of the operations activities and the subsystems or operators by which they are performed. The scenarios are provided as examples of typical operations activities and are not intended to indicate the only sequence of activities that may occur during nominal activities.

#### 3.3.1 Start Up LPGS

LPGS startup activities are performed to boot and power on LPGS hardware, initialize the LPGS UI, and initiate optional periodic monitoring activities.

Step	Subsystem/Operator	Action
1	Sys Opr	Power on/boot workstations and monitors, and log onto operations interface workstation
2	Sys Opr	Start Oracle DBMS
3	Sys Opr	Start up LPGS user interface
4	Sys Opr	Select option to start tasks
5	PCS	Start up LPGS background tasks
6	Prod Opr	Display LPGS event log
7	Analyst	Log onto QA/AA workstation
8	Analyst	Display anomaly main window
9	Analyst	Display anomaly status table

#### 3.3.2 Shut Down LPGS

LPGS shutdown activities are performed to take LPGS hardware off line and terminate all software processes. The LPGS startup scenario must be performed before the LPGS shutdown scenario.

Step	Subsystem/Operator	Action
1	Sys Opr	Display LPGS UI
2	Sys Opr	Display LPGS event log
3	Prod Opr	Select terminate tasks option from menu
4	PCS/DMS/QAS	Terminate current processing
5	Analyst	Exit anomaly windows and log off QA/AA workstation
6	Sys Opr	Exit LPGS UI
7	Sys Opr	Terminate Oracle DBMS
8	Sys Opr	Log off operator workstation
9	Sys Opr	Power off workstations and monitors

## REVIEW

### 3.3.3 Process L1 Product (Nominal End-to-End Processing Flow)

The L1 product processing scenario provides an example of nominal processing of a single L1G product without errors, human intervention, or visual inspection of the product before distribution. The system startup scenario must be performed before commencement of the process L1 product scenario. The following scenario assumes that the produced L1 image is of acceptable quality and that the work order has been set to indicate no visual inspection.

Step	Subsystem/Operator	Action
1	DMS	Poll ECS server for new L1 product generation request
2	DMS	Detect new L1 product generation request
3	DMS	Retrieve and process L1 product generation request
4	DMS	Send L1 product generation response to ECS via ftp
5	DMS	Store information from L1 product generation request in LPGS database
6	DMS	Assess system ingest criteria for L0R product and indicate whether ingest criteria are satisfied
7	DMS	Identify the next product generation request that needs L0R data
8	DMS	Create product request directory
9	DMS	Send start session message to ECS to establish TCP/IP socket connection
10	DMS	Receive start session acknowledgment from ECS indicating successful connection
11	DMS	Send L0R data request to ECS
12	DMS	Receive L0R data request acknowledgment from ECS
13	DMS	Send close session message to terminate TCP/IP socket connection
14	(ECS)	Stage L0R product on ECS disk space
15	(ECS)	Send L0R data availability notice to LPGS via ftp
16	DMS	Poll for new L0R data availability notice
17	DMS	Detect new L0R data availability notice
18	DMS	Retrieve and process L0R data availability notice
19	DMS	Create product request, input, and save directories
20	DMS	Send L0R data availability acknowledgment to ECS via ftp
21	DMS	Retrieve L0R product from ECS via ftp
22	DMS	Verify that the correct L0R product files were received
23	DMS	Catalog L0R product files in database
24	DMS	Send L0R data delivery notice to ECS via ftp
25	DMS	Send L0R data delivery acknowledgment to LPGS via ftp
26	(ECS)	Poll for new L0R data delivery acknowledgment
27	DMS	Detect new L0R data delivery acknowledgment
28	DMS	Retrieve and process L0R data delivery acknowledgment
29	DMS	Update database to indicate that ingest has completed for product generation request

## REVIEW

Step	Subsystem/Operator	Action
30	PCS	Poll database for product generation requests ready for work order processing (L0R ingest completed)
31	PCS	Detect new product generation request ready for work order processing
32	PCS	Generate work order for product generation request. Determine which procedure to use that identifies scripts to run. Determine script parameter values by overriding default values with information provided in product generation request
33	PCS	Create work order directory
34	PCS	Assess resource availability and start work order processing when adequate resources are available
35	PCS	Set up L0R product processing script parameters
36	PCS	Start L0R product processing script
37	DMS	Check data accuracy and generate L0R statistics and consensus PCD and MSCD files
38	DMS	Update database with results
39	PCS	Assess L0R product processing script status and determine that processing continues
40	PCS	Set up L1R processing script parameters
41	PCS	Start L1R processing script
42	RPS	Perform radiometric characterization and correction
43	RPS	Update database with results
44	PCS	Assess L1R processing script status and determine that processing continues
45	PCS	Set up L1R QA script parameters (including thresholds)
46	PCS	Start L1R QA script
47	QAS	Assess results of radiometric characterization and correction
48	QAS	Update database with L1R QA results
49	PCS	Assess L1R QA script status to determine that processing continues
50	PCS	Set up L1G processing script parameters
51	PCS	Start L1G processing script
52	GPS	Perform geometric correction
53	GPS	Update database with results
54	PCS	Assess L1G processing script status and determine that processing continues
55	PCS	Set up L1G QA script
56	PCS	Start L1G QA script
57	QAS	Assess results of geometric correction
58	QAS	Update database with L1G QA results
59	PCS	Assess L1G QA script status to determine that processing continues
60	PCS	Set up formatting script parameters
61	PCS	Start formatting script

## REVIEW

Step	Subsystem/Operator	Action
62	DMS	Format L1G product
63	DMS	Package L1G product
64	DMS	Move product to L1 delivery directory
65	DMS	Check product in L1 delivery directory for completeness
66	PCS	Assess formatting script status to determine that processing continues
67	PCS	Update database to indicate that L1 product is ready for shipment to ECS and that trending data is available for IAS
68	DMS	Place L1 product availability notice in directory on LPGS that is accessible by ECS
69	(ECS)	Poll for new L1 product availability notices
70	(ECS)	Retrieve L1 product via ftp and perform ECS ingest functions
71	(ECS)	Send L1 product availability response to LPGS via ftp
72	DMS	Poll for L1 product availability response
73	DMS	Detect L1 product availability response
74	DMS	Retrieve and process L1 product availability response
75	DMS	Update database to indicate that L1 product has been delivered and product generation request is complete
76	DMS	Update deletion flag for product generation request to indicate that all files associated with request are eligible for deletion

### 3.3.4 Cancel L1 Processing

LPGS activities for canceling L1 processing are performed to terminate processing and resolve all data associated with the canceled request. Requests for canceling L1 processing can be received at any time after the LPGS receives the applicable L1 product generation request and before the LPGS distributes the product to the ECS. The following scenario assumes that L1 processing of the applicable product generation request has been conducted up to Step 53 of the L1 product processing scenario in Section 3.3.3, which assesses the status of L1G processing script results.

Step	Subsystem/Operator	Action
1	(ECS)	Verbally notify LPGS operator of L1 product cancellation
2	Prod Opr/UI	Display current status of applicable product request
3	Prod Opr/UI	Enter L1 product cancellation request
4	UI/DMS	Update cancellation status of product generation request in database to indicate pending cancellation
5	PCS	When L1G processing script completes, assess status of L1G processing script for applicable work order and determine that processing continues
6	PCS	Check cancellation status of product generation request associated with work order

## REVIEW

Step	Subsystem/Operator	Action
7	PCS	Do not initiate execution of L1G quality assessment script or any subsequent scripts that are needed to complete nominal L1G processing
8	PCS	Update work order state in database to indicate that it has been canceled
9	PCS	Update product generation request state in database to indicate that it has been canceled
10	PCS	Update deletion flag for the product generation request to indicate that all files associated with request are eligible for deletion
11	PCS	Update cancellation status for product generation request to indicate that cancellation is complete

### 3.3.5 Retrieval of Characterization Results by IAS

LPGS activities for transferring characterization results to the IAS are performed to provide the IAS with a source of additional L0R and L1 radiometric characterization statistics for use in trending. After the IAS has retrieved the data, the characterization statistics are deleted from the LPGS database. Characterization results are made available to the IAS by the LPGS for products generated from completed work orders. The results are retrieved by the IAS via Oracle SQL\*Net. They are retrieved by the IAS as needed at the discretion of the IAS as indicated in the LPGS/IAS ICD (Requirements Document 6). The DMS, RPS, and GPS tasks generate the trending data and store the data in the database. PCS updates the trending flag for the product generation request to indicate that the data are available for the IAS. The following scenario assumes that characterization results have been written to the LPGS database for completed work orders and that the LPGS startup scenario has been completed successfully.

Step	Subsystem/Operator	Action
1	(IAS)	Connect to LPGS database via SQL*Net
2	(IAS)	Perform query to get changes since last retrieval
3	(IAS)	Retrieve characterization results since last retrieval
4	(IAS)	Update trending flag for all product generation requests whose trending data has been retrieved
5	(IAS)	Disconnect from LPGS database
6	DMS	Poll for product generation requests where trending data has been retrieved by IAS
7	DMS	Delete trending data associated with product generation request
8	DMS	Update trending flag to indicate that trending data have been deleted

## REVIEW

### 3.4 Non-Nominal Operations

Non-nominal operations scenarios are presented in the subsections below. The scenarios are provided as examples of typical non-nominal operations activities and are not intended to indicate the only sequence of activities that may occur.

#### 3.4.1 Analyze Trouble Ticket

LPGS AA activities are performed to identify and resolve anomalies in images produced by the LPGS. The scenario below provides an example of AA in which a trouble ticket is received for an image that has been distributed to a customer, the image problems were reproduced upon reprocessing, and the cause of the problem was found and corrected.

Step	Subsystem/Operator	Action
1	Analyst	Receive notification by e-mail or phone that there is a new trouble ticket assigned to LPGS
2	Analyst/AAS	Access ECS trouble ticket system through UI menu and query for new trouble ticket
3	Analyst/AAS	Access Enter New Anomaly function through UI menu to manually record trouble ticket information into anomalies table
4	Analyst/AAS	If original user product was returned, use ftp to copy product to LPGS disk space (from ECS disk)
5	Analyst/AAS	Access View Image function through UI menu to display original image
6	Analyst	Attempt to verify reported problem. View browse image (View Image) and check metadata (View File)
7	Analyst/AAS	If analyst wishes to rerun request, access Generate Product Request function through UI menu. Store new product request in database
8	DMS	Ingest LOR data required (see steps 6-28 in Section 3.3.3)
9	PCS	Generate work order and set state to "held"
10	Analyst/AAS	Access Modify Work Order function from UI menu to add pauses or to change any parameters
11	Prod Opr/Analyst/AAS	At appropriate time, activate work order
12	Analyst	Monitor and control script processing
13	Analyst	Visually examine product request, work order, work order log, event log, calibration file, QA results, etc.
14	Analyst/AAS	Access View Image function through UI menu to display image
15	Analyst/AAS	If additional run is required, access Generate Work Order function from UI to generate diagnostic work order. This function also generates work order directory
16	Prod Opr/Analyst/AAS	At appropriate time, access Activate Work Order function to activate work order
17	Analyst	Monitor and control script processing
18	Analyst/AAS	Visually inspect results (using View Image)

## REVIEW

Step	Subsystem/Operator	Action
19	Analyst/ AAS	If problem has been resolved, access View/Edit Anomaly function to record resolution in anomalies table
20	Analyst/AAS	Access ECS trouble ticket system through Generate Response function and enter trouble ticket response
21	Analyst/AAS	Manually delete image file provided by ECS in support of trouble ticket

### 3.4.2 Process L1 Product (Non-Nominal)

The non-nominal process L1 product scenario provides an example of the resolution of image anomalies found during routine L1 production processing. The scenario assumes that the nominal process L1 product scenario, in Section 3.3.3, has been conducted up to Step 32. At this point in the L1R processing, quality statistics exceed established thresholds and do not meet quality criteria. This scenario example assumes that the image problem can be resolved by modifying image processing parameters that are specified in the processing work order.

Step	Subsystem/Operator	Action
1	QAS	L1 image fails automated QA
2	PCS	Assess L1R quality script status and update work order state in database to indicate that anomaly has occurred. Send message to operator/analyst indicating work order processing has failed
3	Analyst/AAS	Access Work Order Information function to display failed work order
4.	Analyst/AAS	Access Enter New Anomaly function to add new problem to anomalies table using information available from work order display
5	Analyst/AAS	Access View Event Log function to view event log, Product Request Information function to view product generation request, and Work Order Information function to view work order
6	Analyst/AAS	Access View Files function to view work order log, metadata, and calibration files
7	Analyst/AAS	Access View Image function to view L0R and L1R images
8	Analyst	Develop plan for isolating problem (in this case, suspect systematic problem)
9	Analyst/AAS	Access Generate Work Order function to generate a benchmark work order to verify that LPGS is working properly. Generate work order directory
10	Analyst/AAS	Access Activate Work Order function to activate benchmark work order
11	LPGS	Run benchmark [starting from step 33 of nominal processing flow (Section 3.3.3)]
12	Analyst	Confirm that benchmark run is successful (localized problem rather than systematic one)

## REVIEW

Step	Subsystem/Operator	Action
13	Analyst/AAS	Access Generate Work Order function to create diagnostic work order to process user request with AAS monitoring capabilities. Generate work order directory
14	Analyst/AAS	Access Activate Work Order function to activate and promote diagnostic work order
15	LPGS	Run diagnostic work order [starting from step 33 of nominal processing flow (Section 3.3.3)]
16	Analyst	Monitor and control script processing
17	Analyst	Detect cause of problem that appears correctable and verify necessary processing modifications are documented in log
18	Analyst/AAS	Access Generate Work Order function to create reprocessing work order with necessary corrections. Create work order directory
19	LPGS	Run reprocessing work order and deliver product
20	Analyst/AAS	Access View/Edit Anomaly function to close out anomaly and record resolution in anomalies table

### 3.4.3 Recover From LPGS Failure

LPGS failure recovery activities are performed to isolate and resolve LPGS subsystem failures, notify DAAC management and other affected elements of processing impacts, and continue product processing to the greatest extent possible when failures are isolated to a specific subsystem. The following scenario provides an example of recovery from failures within PCS and assumes that the failure can be resolved by system and production operators without modifying the controlled LPGS configuration.

Step	Subsystem/Operator	Action
1	PCS	Work Order Scheduler (PWS) terminates abnormally
2	PCS	Notify operator of unexpected termination
3	Prod Opr	Display product request processing status
4	Prod Opr	Estimate processing impacts
5	Sys Opr	By voice communications, notify DAAC Manager and EDC DAAC User Services of failure and estimate processing impacts
6	Analyst	Continue QAs and image analysis as much as possible
7	Sys Opr/Prod Opr	Follow operations procedures to resolve and recover from failure
8	Sys Opr	Notify DAAC Manager and EDC DAAC User Services of estimated time to return to full operations
9	Sys Opr/Prod Opr	Resolve failure
10	Prod Opr	Run Recovery function to perform cleanup
11	Sys Opr	Access Restart Tasks function to restart PWS
12	Sys Opr	Notify DAAC Manager and EDC DAAC User Services of return to full operations

# Section 4. Global Libraries

---

## 4.1 Introduction

The LPGS will use global libraries to implement functionality common to the subsystems. The global libraries will also contain non-POSIX-compliant software that needs to be isolated for future software porting purposes.

## 4.2 Design Overview

This section provides an overview of the global library design and a discussion of the considerations used in the design process.

### 4.2.1 Library Overview

As mentioned, the global libraries consist of functions that are shared by the LPGS subsystems or are non-POSIX compliant.

### 4.2.2 Design Considerations

The major assumptions or considerations influencing the design of the global library software are as follows:

- All global library routines will be designed so that the application programmatic interfaces (APIs) are kept to a minimum. On the other hand, the APIs from the global libraries to the operating system will be transparent when the global functions are implemented.
- A global library name will follow standard LPGS naming conventions but will be prefixed with “xxx” to identify it as a global function.
- A global database library name will follow standard LPGS naming conventions but will be prefixed with “xdb” to identify it as a global database function.
- The global functions will be responsible for releasing memory allocated within the global function.

## 4.3 Library Design

### 4.3.1 IAS Global Utilities

The following table identifies the IAS global utilities that are being reused by the LPGS. Descriptions of these modules can be found in the IAS Critical Design Specification (CDS) (Requirements Document 9).

## REVIEW

Function Name	Functional Description
xxx_BuildWOParmODL	Build work order parameter ODL file
xxx_ConnectToDB	Connect to database
xxx_CreateEvent	Create event (called by xxx_PutEvent)
xxx_db_ConnectTo Database	Connect to database (called by xxx_ConnectToDB)
xxx_db_DisconnectFromDatabase	Disconnect from database (called by xxx_DisconnectFromDB)
xxx_db_PutEvent	Write event to database
xxx_DisconnectFromDB	Disconnect from database
xxx_FileLock	Lock/unlock a file
xxx_GetODLField	Get ODL field and convert to proper type
xxx_GetParm	Retrieve an input parameter's value. User provides storage location for value. Function retrieves value from program's input ODL file of parameters and converts value to user-defined data type
xxx_GetTime	Get current time
xxx_Handler	Set up signal handler
xxx_LogError	Report error to system log
xxx_OpenODL	Open ODL file and parse it
xxx_Put_db_Activity	Put activity (called by xxx_PutEvent)
xxx_PutEvent	Write an event to event log in database
xxx_PutTrend	Write program trending data to database
xxx_ReadDAACMetadata	Read DAAC metadata and store in structure
xxx_ReadGEO	Read HDF GEO file and store it in structure
xxx_ReadHDFDirectory	Read HDF directory file and store it in structure
xxx_ReadMetadata	Read EDC and LPS metadata file and store it in structure
xxx_ReadMSCD	Read MSCD file and store it in structure
xxx_ReadPCD	Read PCD file and store it in structure
xxx_ReadSLO	Read SLO file and store it in structure
xxx_Signal	Defines a signal handler to catch all signals
xxx_StartProcess	fork/exec a program
xxx_Unlock	Unlock macro (used by xxx_LogError)
xxx_WriteLock	Write lock macro (used by xxx_LogError)

### 4.3.2 LPGS Global Utilities

The following table identifies the additional global utilities needed for the LPGS.

Function Name	Functional Description
xxx_display_output	Display output to terminal
xxx_get_user_input	Retrieve user input
xxx_proc_cancel_req	Process product cancellation request

## REVIEW

The following presents the module descriptions for the LPGS global utilities.

### **xxx\_display\_output—Display Output for the User**

#### **Parameters:**

xxx\_display\_type : data\_in  
xxx\_display\_data : data\_in  
xxx\_status : control\_out

**Body:** This global library module is called to display output for the user. This module receives the data for the display and brings the data up on the users console.

### **xxx\_get\_user\_input—Get User Input**

#### **Parameters:**

xxx\_status : control\_out  
xxx\_user\_input : data\_out

**Body:** This global library module is called when an LPGS process is notified that it has a user input. This module returns the user input.

### **xxx\_proc\_cancel\_req—Process Cancel Product Request**

#### **Parameters:**

xxx\_cancel\_flag : control\_out  
xxx\_wo\_id : data\_in  
xxx\_prod\_req\_id : data\_in  
xxx\_stat : control\_out

**Body:** This module is invoked to check if the product request has been canceled and if so to process the cancellation.

If the xxx\_prod\_req\_id is set to null this module calls xdb\_get\_prod\_req\_id to get xxx\_prod\_req\_id, the identification of the product request associated with the xxx\_wo\_id. It then calls xdb\_check\_cancel\_stat to see if a request to cancel the product request identified by xxx\_prod\_req\_id has been received and is pending.

If the product request needs to be canceled, this module calls xxx\_PutEvent to put the event into the database. Then, it calls the pdb\_UpdateWOState to update the work order state to canceled. It calls xdb\_update\_prod\_req\_state to set the product request's state to canceled. The xdb\_upd\_cancel\_flag module is then called to set the current product request's cancel state to done. Finally, the xdb\_upd\_prod\_req\_del\_flag module is called to set the current product request's delete flag to deletable and xxx\_cancel\_flag is set to indicate that the cancellation was done.

## REVIEW

Otherwise the product request is not canceled and xxx\_cancel\_flag is set to indicate that the cancellation was not done.

Errors encountered in the module are returned in xxx\_stat.

### 4.3.3 IAS Global Database Utilities

The following table identifies the IAS global database utilities that are being reused by the LPGS. Descriptions of these modules can be found in the IAS CDS (Requirements Reference 9).

Function Name	Functional Description
xdb_GetCPFValidation	Get CPF validation information
xdb_GetDBMSLogin	Get DBMS login information
xdb_GetIASConfig	Get IAS configuration information
xdb_GetIngestParameters	Get LOR ingest parameters
xdb_GetScriptParms	Get work order script parameters
xdb_InsertORTrending	Insert LOR trending
xdb_PutEventLog	Write event to log

### 4.3.4 LPGS Global Database Utilities

The following table identifies the additional global database utilities to be developed for the LPGS.

Function Name	Functional Description
xdb_check_cancel_stat	Check product cancellation status
xdb_create_wo_log	Create work order log
xdb_get_prod_dir	Get product request directory
xdb_get_prod_req	Get product request
xdb_get_prod_req_id	Get product request identifier
xdb_get_task_config	Get task configuration parameters
xdb_get_time_period	Get timer parameter
xdb_insert_wo	Insert new work order
xdb_store_wo_dir	Update work order directory
xdb_upd_cancel_flag	Update product request cancellation status
xdb_upd_prod_req_del_flag	Update product request delete flag
xdb_update_pr	Update product request
xdb_update_prod_req_state	Update product request state

The following are the module descriptions for the LPGS database utilities.

## REVIEW

### **xdb\_check\_cancel\_stat—Check Cancel Status**

#### **Parameters:**

xxx\_prod\_req\_id : data\_in  
xxx\_cancel\_flag : control\_out  
xdb\_stat : control\_out

**Body:** This module will check the cancellation status field in the Product Request table for this xxx\_prod\_req\_id to determine if the product request has been canceled. If so the xxx\_cancel\_flag is set to TRUE and the module terminates. xdb\_stat is returned to indicate successful or unsuccessful completion.

### **xdb\_create\_wo\_log—Create Work Order Log**

#### **Parameters:**

xdb\_stat : control\_out  
xxx\_wo\_id : data\_in

**Body:** This module uses xxx\_wo\_id to create an empty work order log in the work order's temporary directory and stores the name of the log file into the Work Orders table.

Errors encountered by the module in accessing the database or creating the log are returned in xdb\_stat.

### **xdb\_get\_prod\_dir—Get Product Directory Name**

#### **Parameters:**

xxx\_prod\_req\_dir : data\_out  
xxx\_status : control\_out  
xxx\_prod\_req\_id : data\_in

**Body:** This module accesses the product request database table to get the name of the product request directory for the specified product. This module returns the name of the product request directory.

**NOTE:** The following is the directory structure for the product requests:

- product\_request\_dir
  - input\_directory
    - LOR Product files in this directory
  - save\_directory
    - Files destroyed during processing are saved here
  - wo1\_directory

## REVIEW

L1R & L1G Output files in this directory  
wo2\_directory  
L1R & L1G Output files in this directory  
... Each time a work order is run, a new work order directory is created

### **xdb\_get\_prod\_req—Get The Product Request From The Database**

#### **Parameters:**

xdb\_stat : control\_out  
xxx\_prod\_req\_id : data\_in  
xxx\_prod\_req : data\_out

**Body:** This module gets the specified product request from the database. If the product request is not found, this module returns an error.

### **xdb\_get\_prod\_req\_id—Get Product Request ID**

#### **Parameters:**

xxx\_prod\_req\_id : data\_out  
xxx\_wo\_id : data\_in  
xdb\_stat : control\_out

**Body:** This module reads the work order table using xxx\_wo\_id to retrieve the xxx\_prod\_req\_id associated with the work order. It returns xdb\_stat to indicate success or failure.

### **xdb\_get\_task\_config—Get Task Configuration Variables**

#### **Parameters:**

xdb\_stat : control\_out  
xxx\_task\_id : data\_in  
xxx\_task\_config : data\_out

**Body:** This module returns a structure, xxx\_task\_config, where each element is a column from the Sys\_Params table in the LPGS database that matches the specified xxx\_task\_id. These elements are used for setting task-specific global parameters. Errors encountered in the module are returned in xdb\_stat.

## REVIEW

### **xdb\_get\_time\_period—Get Time Period From The Database**

#### **Parameters:**

xdb\_stat : control\_out  
xdb\_event\_type : data\_in  
xdb\_time\_period : data\_out

**Body:** The operator can specify the time period between certain events within the LPGS system. This module gets the time period for the event from the database.

The following are the defined timeouts:

TASK	EVENT
DGR	Time period for collecting the LPGS Statistics
DGR	Time period for IAS interface
DIL	Time period to wait for an ACK message from ECS after sending a message on a socket
DIL	Time period for polling the DAN and DDA directories for files
DIL	Time period between requests for Level 0 data
DXL	Time period between checking the database for product requests that need a PDR file created
DXL	Time period for polling for L1 Product Available Response (PDRD or PAN) files from ECS

### **xdb\_insert\_wo—Insert Work Order**

#### **Parameters:**

xxx\_wo\_id : data\_out  
xdb\_wo\_state : data\_in  
xxx\_prod\_req\_id : data\_in  
xxx\_wo\_proc\_id : data\_in  
xdb\_stat : control\_out

**Body:** This module first gets the next value of xxx\_wo\_id from the LPGS database. It then inserts a new work order into the Work Orders table using the information specified by xxx\_wo\_proc\_id, xxx\_prod\_req\_id, xxx\_wo\_id, and xdb\_wo\_state. Database access status is returned to the calling module in xdb\_stat.

## REVIEW

### **xdb\_store\_wo\_dir—Store WO Directory**

#### **Parameters:**

xxx\_wo\_id : data\_in  
xxx\_wo\_dir : data\_in  
xdb\_stat : control\_out

**Body:** This module uses xxx\_wo\_id to store xxx\_wo\_dir, the location of the work order's root directory, into the Work Orders table. Errors encountered by the module in accessing the database are returned in xdb\_stat.

### **xdb\_upd\_cancel\_flag—Update Cancel Flag**

#### **Parameters:**

xxx\_prod\_req\_id : data\_in  
xxx\_cancel\_state : control\_in  
xdb\_stat : control\_out

**Body:** This module updates the cancel flag field in the product request table using xxx\_prod\_req\_id and xxx\_cancel\_state. It returns xdb\_stat to indicate successful or unsuccessful completion.

### **xdb\_upd\_prod\_req\_del\_flag—Update Product Request Delete Flag**

#### **Parameters:**

xxx\_prod\_req\_id : data\_in  
xxx\_del\_state : control\_in  
xdb\_stat : control\_out

**Body:** This module updates the delete flag in the product request table for a product request identified by xxx\_prod\_req\_id. It updates the status to xxx\_del\_state and returns xdb\_stat to indicate successful or unsuccessful completion.

### **xdb\_update\_pr—Update Product Request in DB**

#### **Parameters:**

xdb\_stat : control\_out  
ddb\_del\_pr\_dir : data\_in

**Body:** This module updates the product request entities in the LPGS database to reflect the deletion of L0 and L1 products.

## REVIEW

### **xdb\_update\_prod\_req\_state—Update Product Request State**

#### **Parameters:**

xxx\_prod\_req\_state : data\_in

xxx\_prod\_req\_id : data\_in

xdb\_stat : control\_out

**Body:** This module updates the product request state associated with xxx\_prod\_req\_id in the LPGS database to the value specified by xxx\_prod\_req\_state. The success or failure of the update is returned in xdb\_stat.

# Section 5. Process Control Subsystem

---

## 5.1 Introduction

The PCS controls LPGS production planning and processing. It processes product generation requests and sets up, monitors the status of, and controls the processing of LPGS work orders. The PCS provides processing status in response to operator requests.

## 5.2 Design Overview

This section provides an overview of the PCS software design. It presents the relationships between the PCS and the other LPGS subsystems. It also discusses the considerations and assumptions used in the design process.

### 5.2.1 Subsystem Software Overview

Figure 5-1 contains the PCS context diagram.

The PCS creates a work order for a product generation request after the DMS has successfully ingested the LOR data. It selects the appropriate work order procedure to be run from the set of available procedures maintained in the database. The UI populates the database with the list of procedures. The work order procedure identifies an ordered list of scripts that need to be run to produce the requested product. The procedure also supplies the default parameters for each script. The PCS uses the options specified in the product generation request to override the default parameters. The PCS stores the work order in the database.

The PCS starts work order processing when the system resources are available for processing. The work order specifies the order in which the scripts are to be run. The work order also indicates when processing is to be halted after a script completes; this allows for human intervention, if necessary. The scripts run DMS, QAS, RPS, and GPS application programs. The PCS initiates execution of the work order scripts in sequence until the work order completes or it encounters a halt condition. The PCS provides required parameters to each script. If work order processing completes successfully, the PCS updates the product request state in the database to notify the DMS that the product is ready for transfer to ECS. In the case of a failure, the PCS notifies the AAS analyst of the problem by updating the work order state in the database to indicate that an anomaly has occurred. For a work order that has been halted, the PCS waits for the operator to resume it using a UI tool. The PCS periodically polls the database looking for the operator's response. The PCS resumes work order processing with the next script when system resources are available.

The PCS also performs cancellation of product generation requests that are in the work order processing phase. The PCS terminates processing of the corresponding work order at a convenient point (e.g., between scripts). It updates the database to indicate that the work order and the product generation request have been canceled, and that the files associated with the request are eligible for deletion.

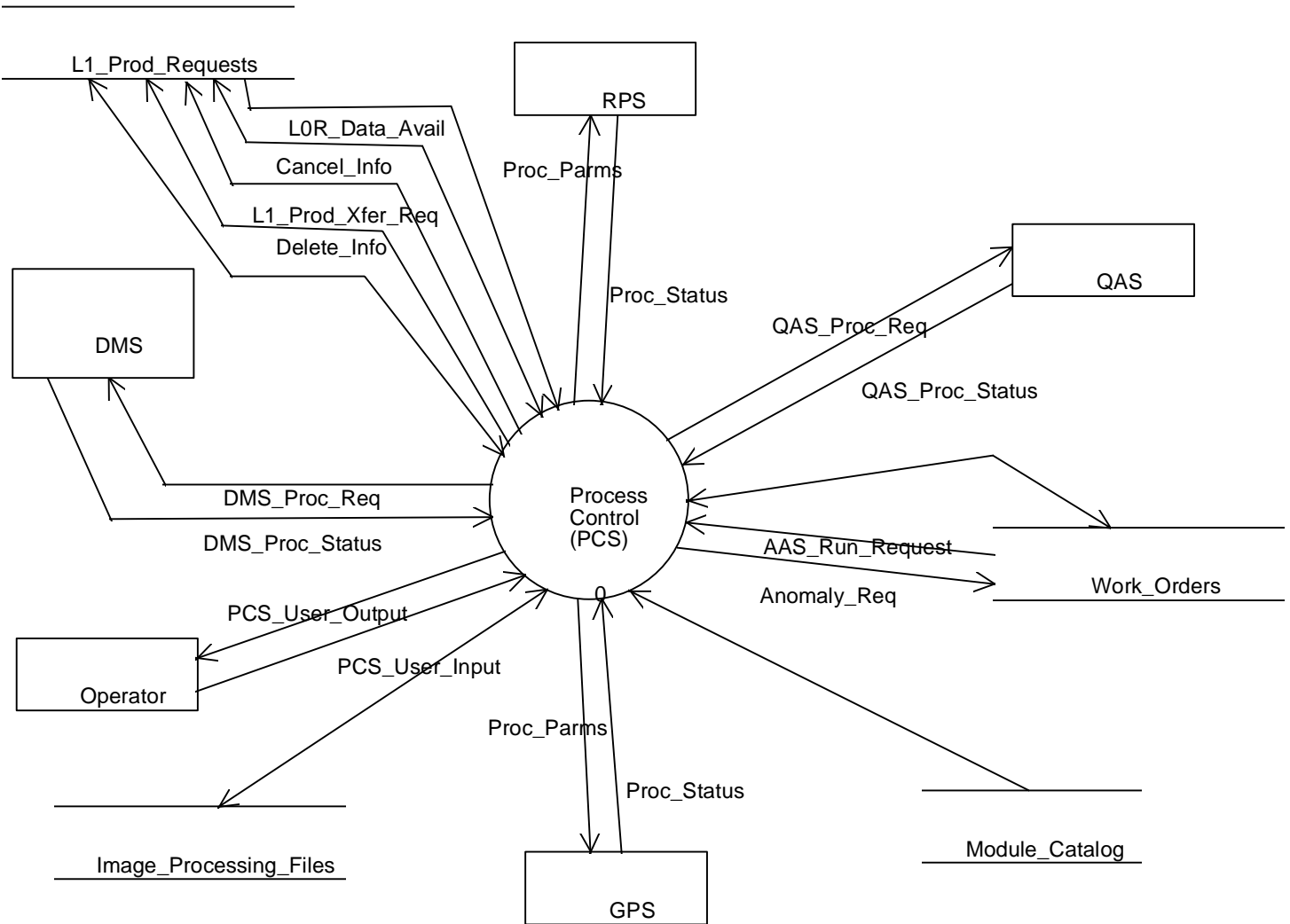


Figure 5-1. PCS Context Diagram

## REVIEW

### 5.2.2 Design Considerations

The LPGS PCS design approach was to reuse the IAS PCS design where the functions are similar.

### 5.2.3 Design Assumptions

The following assumptions were made in designing the PCS:

- **Order of Processing**—The PCS processes work orders on a first-in, first-out (FIFO) basis except when the operator promotes a work order. When the required system resources are available, the PCS selects the oldest promoted work order or, if there are no promoted work orders, the oldest work order to process next.
- **Parallel Processing of Work Orders**—The PCS supports concurrent processing of work orders. Operator-controlled parameters are used to restrict the number of in-progress and actively executing work orders. The parameters are used to control the flow through the production pipeline.
- **Cancellation Points**—The PCS can process a product cancellation request at various points between generation of the work order and completion of work order processing. If the work order is currently executing, the PCS waits for the script to terminate before processing the cancellation.
- **Visual Quality Assessment**—The PCS supports visual quality assessment through the use of halts. The QAS analyst performs the visual assessment and then resumes the work order using a UI tool. An operator-controlled parameter specifies whether visual quality assessments are to be performed.

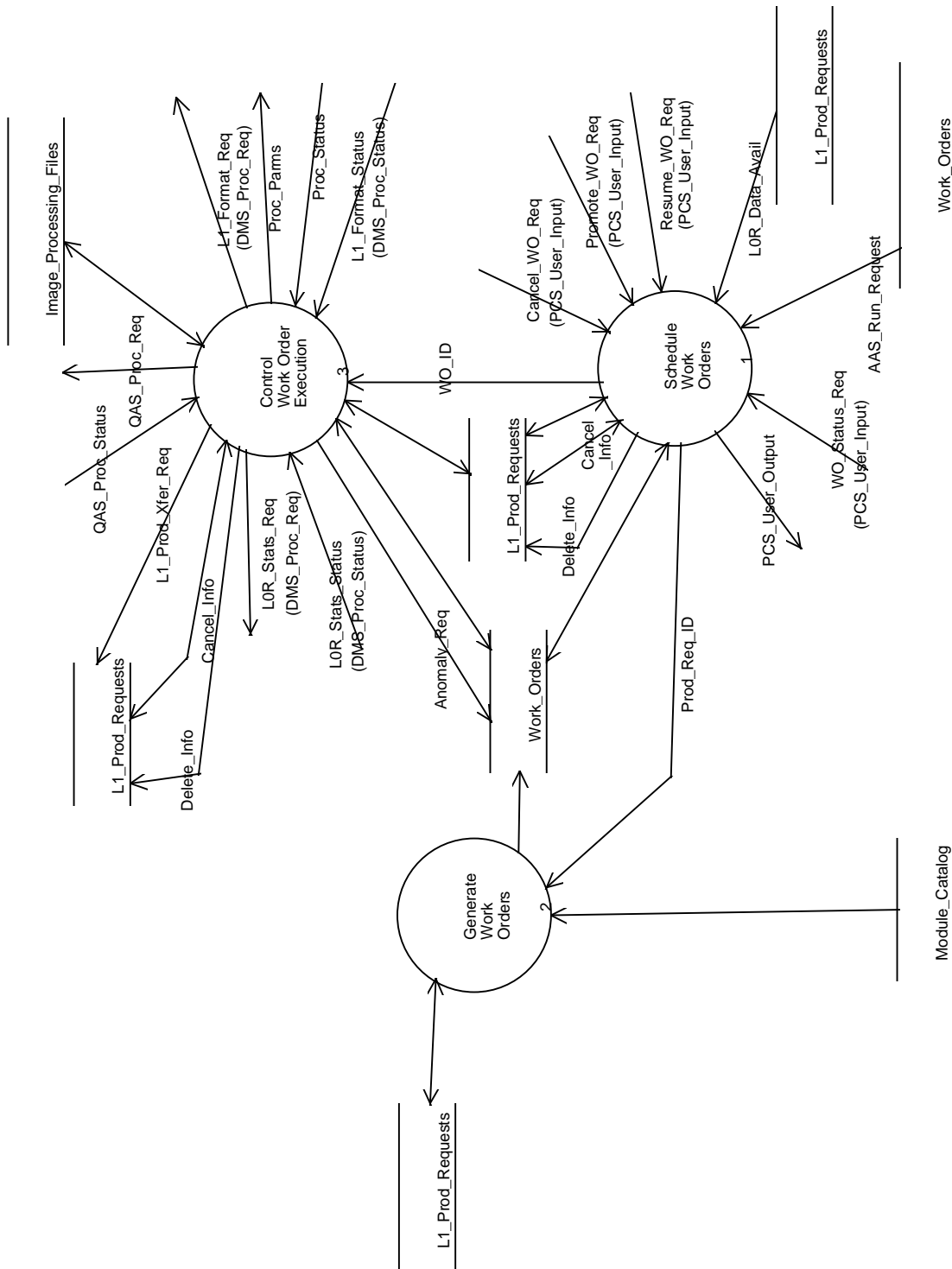
## 5.3 Subsystem Design

This section provides a description of the PCS software architecture selected to implement the PCS design. Figure 5-2 shows the PCS software architecture. The PCS is composed of the System Initialization/Termination (PSI), the Work Order Generator (PWG), the Work Order Scheduler (PWS), and the Work Order Controller (PWC) tasks. The PSI and PWS tasks run in the background at all times.

The PSI task starts and monitors all LPGS background tasks. It is also responsible for notifying the background tasks of system shut down.

The PWS task periodically polls the database to determine when there is work to do. It polls the Work Order table to find work orders that need to be started or resumed. PWS nominally starts work orders in the order that they were received. A UI tool allows the operator to change the order of work orders in the schedule. To initiate or resume work order processing, the PWS task performs a fork and exec of the PWC task. The PWS uses a system level parameter to control the number of PWC tasks active at one time. PWS also polls the Product Request table to

## REVIEW



**Figure 5-2. PCS Level 0 DFD**

## REVIEW

identify product generation requests that are ready for work order processing (LOR ingest has completed successfully). It performs a fork and exec of the PWG task. The PWS uses a system level parameter to control the number of PWG tasks active at one time.

The PWC task controls and monitors the execution of work order scripts. It builds the environment of each script (creating the ODL parameter file) and performs a fork and exec of the script. A script runs one or more DMS, RPS, GPS, or QAS programs. PWC waits for script completion. It continues processing scripts until it reaches the last script or encounters a halt condition.

### 5.3.1 System Initialization/Termination (PSI) Task

The System Initialization/Termination task is responsible for starting up and shutting down the LPGS background tasks. It retrieves the LPGS task configuration information from the database and starts each task. Then PSI monitors for abnormal termination of any of the background tasks and waits for any user directives. When PSI receives a shut down directive from the UI, it notifies all of the LPGS background tasks of the shut down. This task is a 100 percent reuse of the IAS PSI task.

### 5.3.2 Work Order Generator (PWG) Task

This subsection describes the PWG task software.

#### 5.3.2.1 Task Overview

The Work Order Generator task is responsible for automatic creation of work orders. It generates the work order for the product generation request specified in the input parameter passed to it by the PWS task. It retrieves the product generation request options from the database and uses this information to determine which procedure is to be used to process the request. PWG retrieves the procedure information including the ordered list of scripts and the default parameters for each script from the database. It takes the procedure information and uses the product generation request options to create the work order. The request options are used to override the default values of the parameters that are used in processing. Once the work order is inserted into the database, PWG creates the work order directory and then exits.

#### 5.3.2.2 Initialization

The PWG task initialization consists of connecting to the database and retrieving the product request identifier supplied as an input parameter to the task.

#### 5.3.2.3 Normal Operation

The PWG task generates the work order for the product generation request that is specified in the input parameter supplied to it by PWS. It first calls `pwg_init` to initialize the task. Then it calls `pwg_gen_wo` to create the work order for this request. PWG creates the work order directory by calling `pwg_gen_wo_dir`. The product request state is updated to indicate that the

## REVIEW

work order processing has begun by calling `xdb_update_prod_req_state`. Finally, PWG calls `xxx_DisconnectFromDB` to disconnect from the database prior to terminating the task.

### 5.3.2.4 Error Handling

The Work Order Generator reports any errors encountered to the system log file and/or the event log and exits gracefully when it is unable to continue processing.

### 5.3.2.5 Design

This subsection presents the design of the PWG task. Figure 5-3 shows the structure chart for the Work Order Generator task. The module specifications for the PWG task are provided (in alphabetical order) below.

#### **pdb\_get\_req\_info—Get Product Request Info**

##### **Parameters:**

`xxx_prod_req_id` : data\_in  
`pdb_stat` : control\_out  
`pwg_prod_req_info` : data\_out

**Body:** This module uses `xxx_prod_req_id` to retrieve `pwg_prod_req_info` for a specific product request from the LPGS database. The module returns database status in `pdb_stat`.

#### **pdb\_get\_script\_params—Get Script Parameters**

##### **Parameters:**

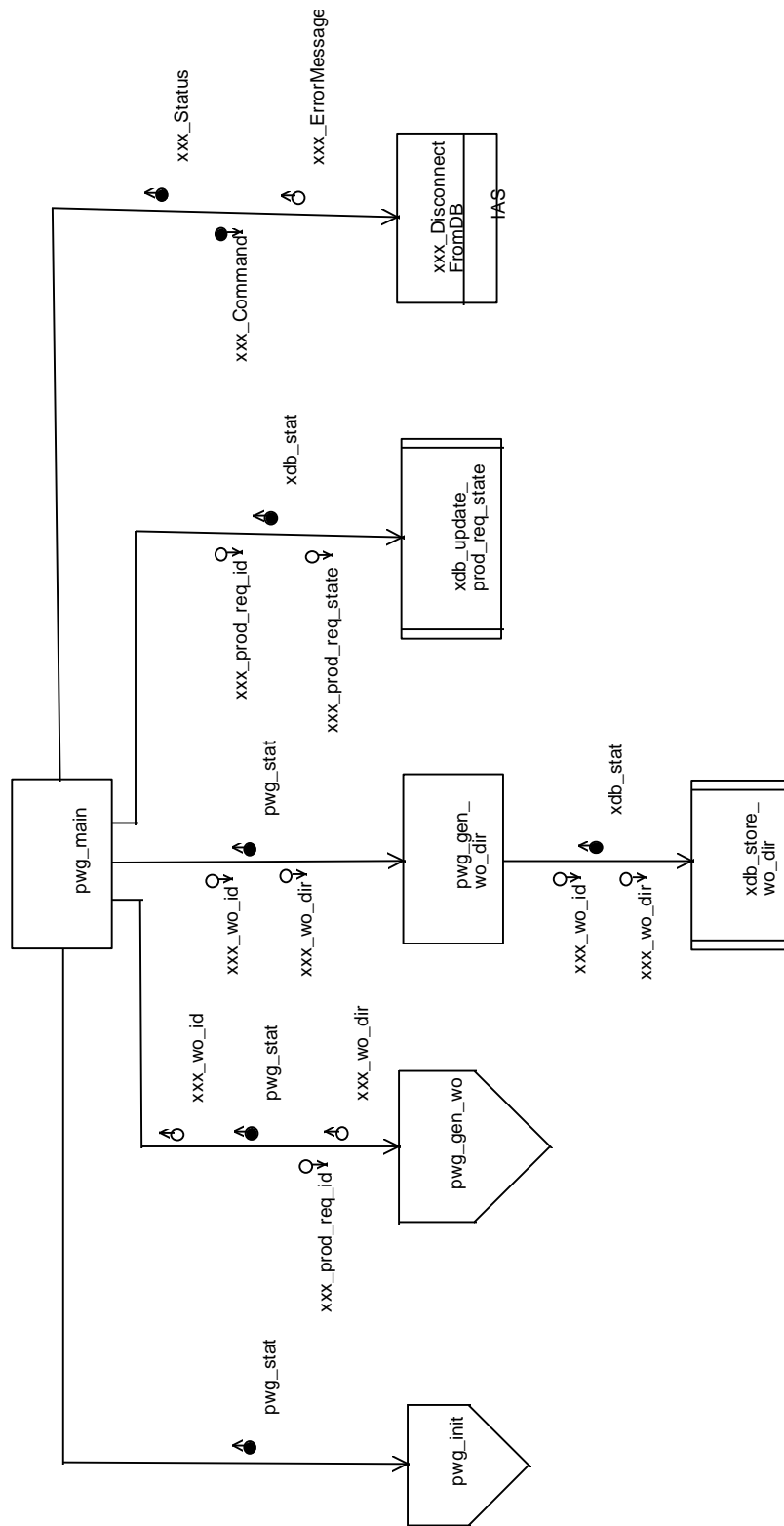
`pwg_prod_req_info` : data\_in  
`xxx_script_id` : data\_in  
`xxx_wo_proc_id` : data\_in  
`xxx_wo_id` : data\_in  
`pdb_stat` : control\_out

**Body:** This module sets up the values to be used for all occurrences of each parameter used in the script specified by `xxx_script_id` for the work order associated with `xxx_wo_id`.

The module does so by looping through each occurrence of each parameter in the specified script. For each occurrence it first uses `xxx_script_id`, `xxx_wo_proc_id`, and the name and occurrence number of the parameter to get the parameter's value from the Default Parameters table. It then checks whether the value needs to be overridden by a corresponding value for the parameter specified in `pwg_prod_req_info` before inserting the resultant parameter value into the Work Order Parameters table.

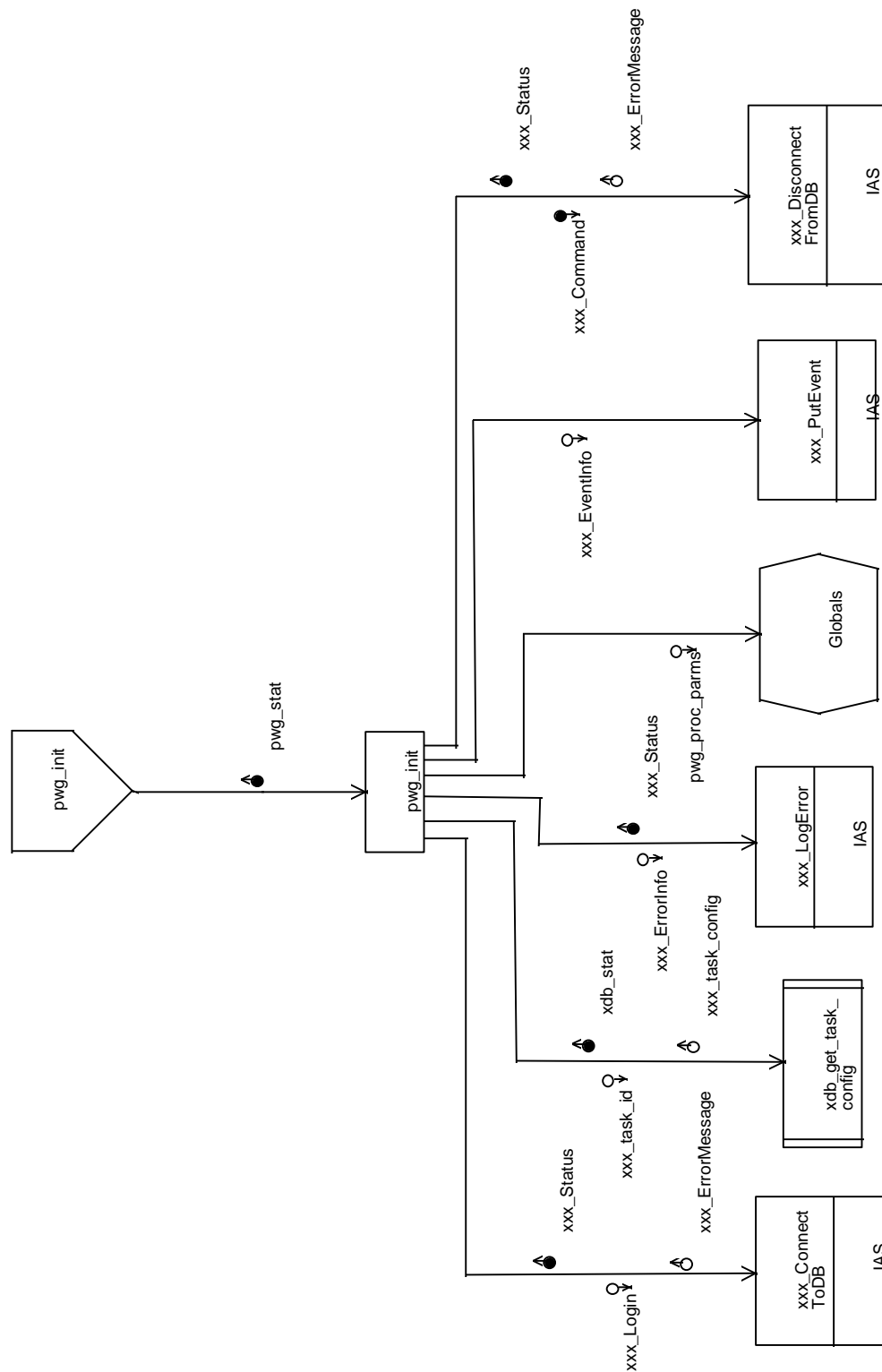
Database access status is returned to the calling module in `pdb_stat`.

## REVIEW



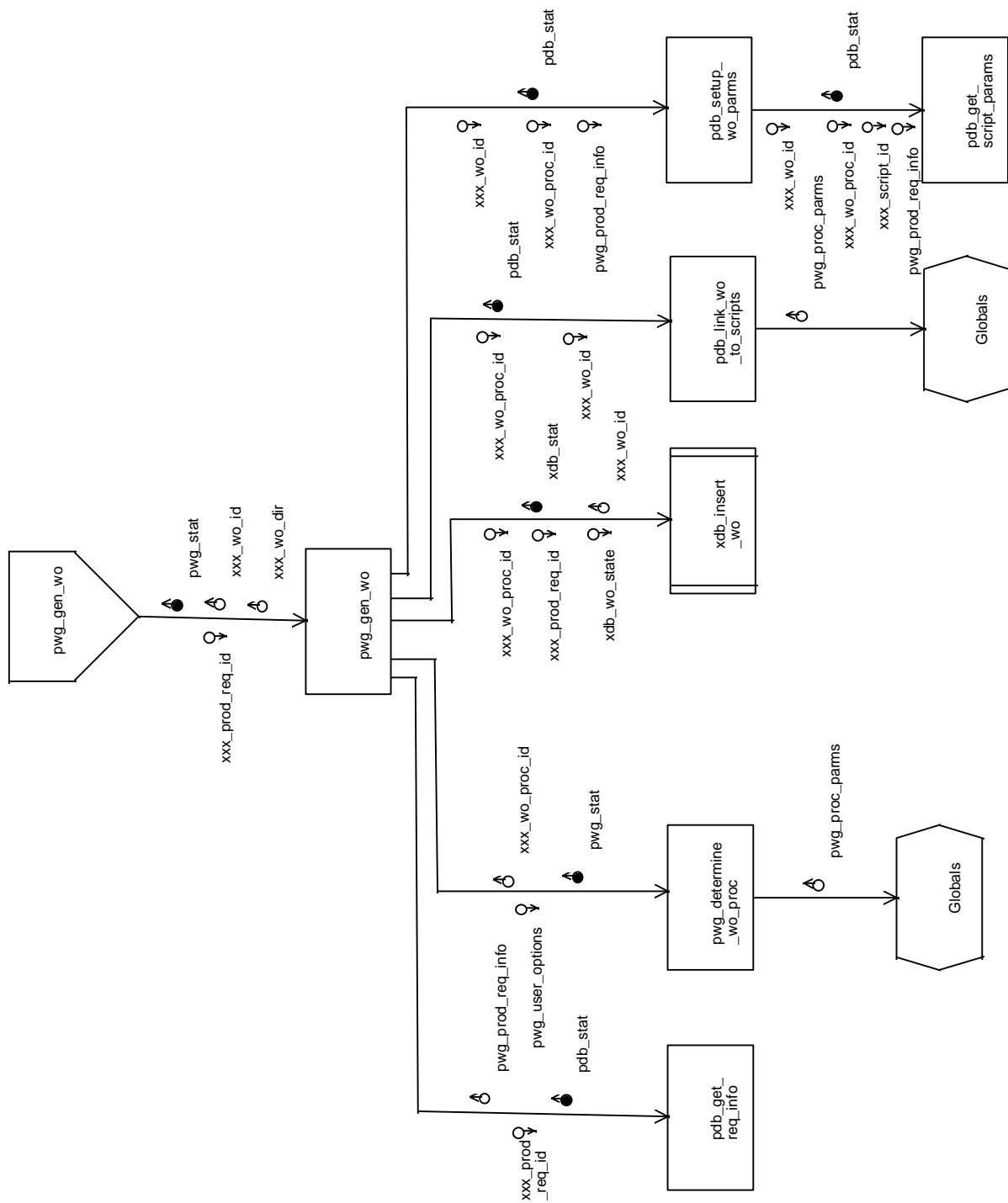
**Figure 5-3. PWG Structure Chart (1 of 3)**

## REVIEW



**Figure 5-3. PWG Structure Chart (2 of 3)**

## REVIEW



**Figure 5-3. PWG Structure Chart (3 of 3)**

## REVIEW

### **pdb\_link\_wo\_to\_scripts—Link Work Order to Scripts**

#### **Parameters:**

xxx\_wo\_id : data\_in

xxx\_wo\_proc\_id : data\_in

pdb\_stat : control\_out

**Body:** This module links the script information associated with the procedure specified by xxx\_wo\_proc\_id with the work order specified by xxx\_wo\_id by using the information in the Procedures table.

The module loops through each script in the procedure. As it does so, it extracts script information from the Procedures table, modifies it if necessary based on the processing parameters specified by pwg\_proc\_parms, and inserts the resultant information into the Work Order Scripts table.

Database access status is returned to the calling module in pdb\_stat.

### **pdb\_setup\_wo\_parms—Set Up Work Order Parameters**

#### **Parameters:**

pwg\_prod\_req\_info : data\_in

xxx\_wo\_proc\_id : data\_in

xxx\_wo\_id : data\_in

pdb\_stat : control\_out

**Body:** This module sets up the parameters associated with each script previously linked to the work order specified by xxx\_wo\_id.

### **pwg\_determine\_wo\_proc—Determine Work Order Procedure ID**

#### **Parameters:**

pwg\_user\_options : data\_in

pwg\_stat : control\_out

xxx\_wo\_proc\_id : data\_out

**Body:** This module uses pwg\_user\_options and pwg\_proc\_parms to determine which canned procedure to use in processing a particular product request. That information is returned as xxx\_wo\_proc\_id. Status is returned to the calling module in pwg\_stat.

## REVIEW

### **pwg\_gen\_wo—Generate Work Order**

#### **Parameters:**

xxx\_wo\_dir : data\_out  
xxx\_prod\_req\_id : data\_in  
pwg\_stat : control\_out  
xxx\_wo\_id : data\_out

**Body:** This module generates a new work order for the nominal processing of the L1 product request specified by xxx\_prod\_req\_id and stores the information associated with the work order in the LPGS database. In doing so, it identifies the sequence of scripts to be run for the work order and the parameters to be used with each script. When it is done, it returns xxx\_wo\_id, the unique identifier of the work order, xxx\_wo\_dir, the location of the work order's root directory, and pwg\_stat, which indicates whether the work order was generated successfully or not.

Fatal errors encountered during processing cause the module to exit prematurely and return an error status through pwg\_stat to the calling module.

#### **PDL:**

```
CALL pdb_get_req_info using xxx_prod_req_id to get the user-specified options and
parameters and the root directory of the current product request from the
Product Requests table
CALL pwg_determine_wo_proc using the user-specified options to identify the
procedure to use for processing the current work order from information in the
Procedures table
CALL xdb_insert_wo using xxx_prod_req_id and xxx_wo_id to insert a record for the
new work order into the Work Orders table
GENERATE xxx_wo_dir, the location of the current work order's directory, from the
root directory of its product request and xxx_wo_id
CALL pdb_link_wo_to_scripts using the identified procedure to get the script
information from the Procedures table, modify it using globally-specified task
parameters, and insert it into the Work Order Scripts table using xxx_wo_id
CALL pdb_setup_wo_parms using xxx_wo_id and the procedure id to determine the
parameters to use for each script in the work order based on the default values in the
Default Parameters table and the values specified by the user, and store them in the
Work Order Parameters table
IF an error is encountered in a database module THEN
    CALL xxx_LogError to write the error message to the LPGS log
    CALL xxx_PutEvent to record the error message in the Event table
    RETURN pwg_stat with error
ENDIF
IF an error is encountered in a PWG module THEN
    RETURN pwg_stat with error
ENDIF
```

## REVIEW

RETURN pwg\_stat with success

### **pwg\_gen\_wo\_dir—Generate Work Order Directory**

#### **Parameters:**

xxx\_wo\_dir : data\_in

xxx\_wo\_id : data\_in

pwg\_stat : control\_out

**Body:** This module generates the intermediate directory needed for processing a work order using xxx\_wo\_dir, the location of the work order's root directory. When it is done, the module calls xdb\_store\_wo\_dir to store the directory information into the Work Orders table. Errors encountered by the module in generating the directory or accessing the database are returned in pwg\_stat.

### **pwg\_init—PWG Initialization**

#### **Parameters:**

pwg\_stat : control\_out

**Body:** This module is responsible for initializing the PWG task. Its main functions are to connect the task to the LPGS database and to get task configuration parameters from the database. It returns pwg\_stat to indicate whether it was successful or not.

### **pwg\_main—Control Work Order Generation**

**Parameters:** xxx\_prod\_req\_id is passed as a parameter to the PWG script.

**Body:** This module receives xxx\_prod\_req\_id, the identification of a product request, as an argument when it is spawned by the PWS task. It first calls on pwg\_init to initialize the task. It then uses xxx\_prod\_req\_id to invoke pwg\_gen\_wo, which generates a new nominal work order for that request and stores its information in the LPGS database.

Next, it calls pwg\_gen\_wo\_dir to generate the subdirectory in which the files to be accessed in processing the work order will reside. Then it calls on xdb\_update\_prod\_req\_state to set the state of the product request in the Product Requests table to indicate that processing of the request has started. Finally, it invokes xxx\_DisconnectFromDB to disconnect the PWG task from the LPGS database before exiting.

Fatal errors encountered during processing cause the task to exit prematurely and return an error code that can be checked by the task's parent.

#### **PDL:**

CALL pwg\_init to initialize the task

CALL pwg\_gen\_wo to generate a work order from the information in the database associated with xxx\_prod\_req\_id and link the work order to the scripts and program parameters that are to be used in processing it

## REVIEW

```
CALL pwg_gen_wo_dir to create the directories to be used by the work order during
    L1 processing
CALL xdb_update_prod_req_state to update the state of the product request to
    indicate that processing of the product request is underway
IF a database error is encountered THEN
    CALL xxx_LogError to write the error message to the LPGS log
    CALL xxx_PutEvent to record the error message in the Event table
    CALL exit with an error termination code that can be checked by the task's
        parent
ENDIF
CALL xxx_DisconnectFromDB to disconnect the task from the LPGS database
IF an error is encountered in a pwg or xxx module THEN
    CALL exit with an error termination code that can be checked by the task's
        parent
ENDIF
CALL exit with a success termination code that can be checked by the task's parent
```

### 5.3.3 Work Order Scheduler (PWS) Task

This subsection describes the PWS task software.

#### 5.3.3.1 Task Overview

The Work Order Scheduler task is responsible for starting up or resuming work order processing and for initiating generation of new work orders. It polls the database for work orders ready to be started/resumed. When there are sufficient disk and CPU resources, it checks to see if the product request or work order has been canceled and processes the cancellation if necessary. Otherwise it performs a fork and exec of the PWC task to start execution of the work order. It also polls the database for product requests whose input data has been staged. It performs a fork and exec of a PWG task for each product request once the LOR ingest has completed.

#### 5.3.3.2 Initialization

The PWS task initialization consists of connecting to the database, setting up a signal catcher for the termination message, and retrieving any task parameters from the database.

#### 5.3.3.3 Normal Operation

PWS performs in a loop until it receives a termination message from the PSI task. It periodically polls the database to determine whether there is work to do. The polling frequency is controlled by a system parameter. PWS calls pws\_init to initialize the task and then begins its loop. First it calls pws\_chk\_child\_term to determine whether any spawned PWC tasks have completed and to adjust the count of the number of actively executing PWCs. Then it invokes pws\_proc\_resumable\_wo to continue processing any work orders that can be resumed. PWS calls pws\_proc\_l0ready\_wo to start any new work orders if there are sufficient system resources. There is a system parameter that is used to limit the number of work orders that are executing (i.e., number of PWC tasks running). Next PWS calls pws\_chk\_child\_term to determine whether

## REVIEW

any spawned PWG tasks have completed and to adjust the count of the number of actively executing PWGs. Then it calls `pws_create_work_orders` to generate work orders for any product requests where the input data has been staged. There is another system parameter that controls the number of work orders that can be generated at one time (i.e., number of PWG tasks running).

PWS calls `pws_term` when it receives a termination message from the PSI task.

### 5.3.3.4 Error Handling

The Work Order Scheduler reports any errors encountered to the system log file and/or the event log and exits gracefully when it is unable to continue processing.

### 5.3.3.5 Design

This subsection presents the design of the PWS task. Figure 5-4 shows the structure chart for the Work Order Scheduler task. The module specifications for the PWS task are provided (in alphabetical order below).

#### **`pdb_get_l0ready_prod_req`—Get Next L0Ready Product Request**

##### **Parameters:**

`xxx_prod_req_id` : data\_out

`pdb_stat` : control\_out

**Body:** This module gets `xxx_prod_req_id`, the request id of the next product request for which a new work order can be generated, from the Product Requests table. This will be the earliest product request for which L0 data is available on LPGS. Errors encountered in the module are returned in `pdb_stat`.

#### **`pdb_get_next_resumable_wo`—Get Next Resumable Work Order**

##### **Parameters:**

`xxx_wo_id` : data\_out

`pdb_stat` : control\_out

**Body:** This module gets `xxx_wo_id`, the identification of the next work order which can be resumed following an earlier interruption caused by a planned halt in processing or a fatal error. Normally this will be the earliest work order in the resumable state that has been interrupted unless another work order in the same state has been promoted to take its place.

Errors encountered in the module are returned in `pdb_stat`.

REVIEW

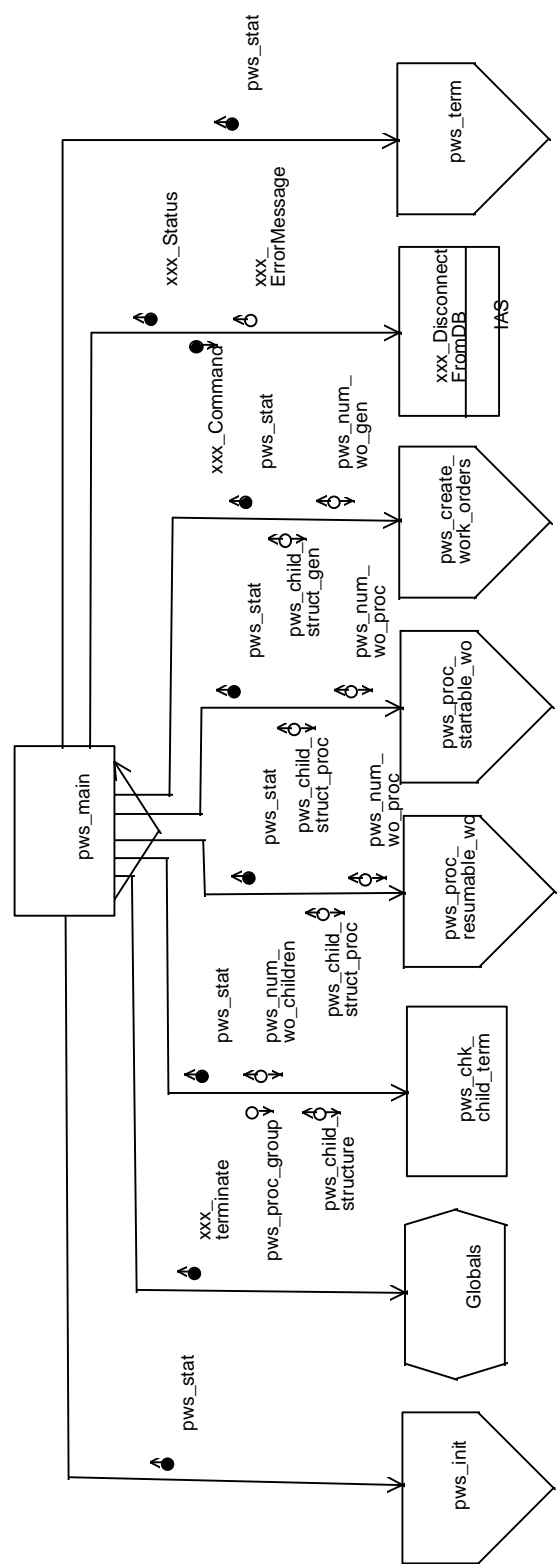


Figure 5-4. PWS Structure Chart (1 of 7)

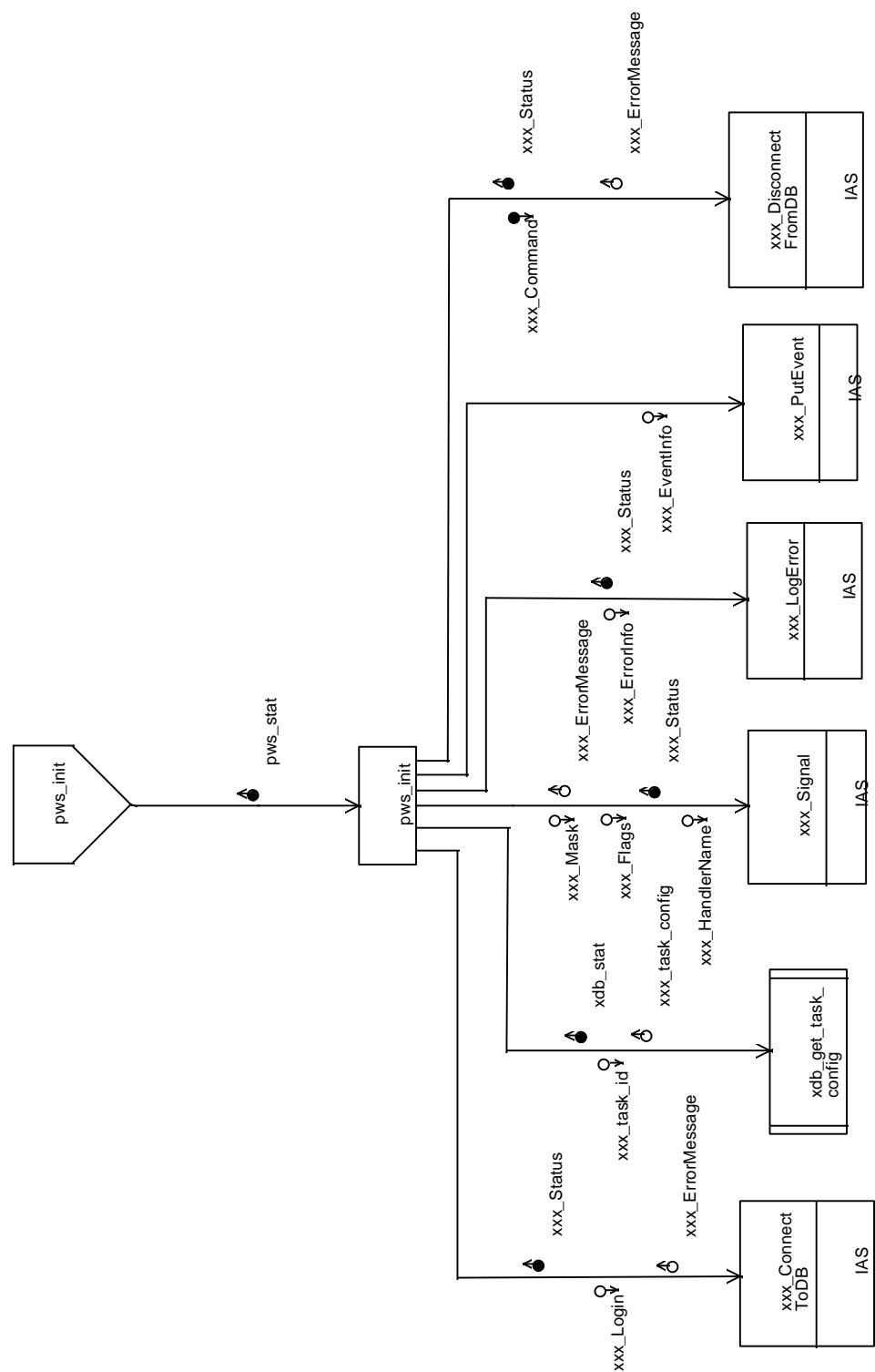


Figure 5-4. PWS Structure Chart (2 of 7)

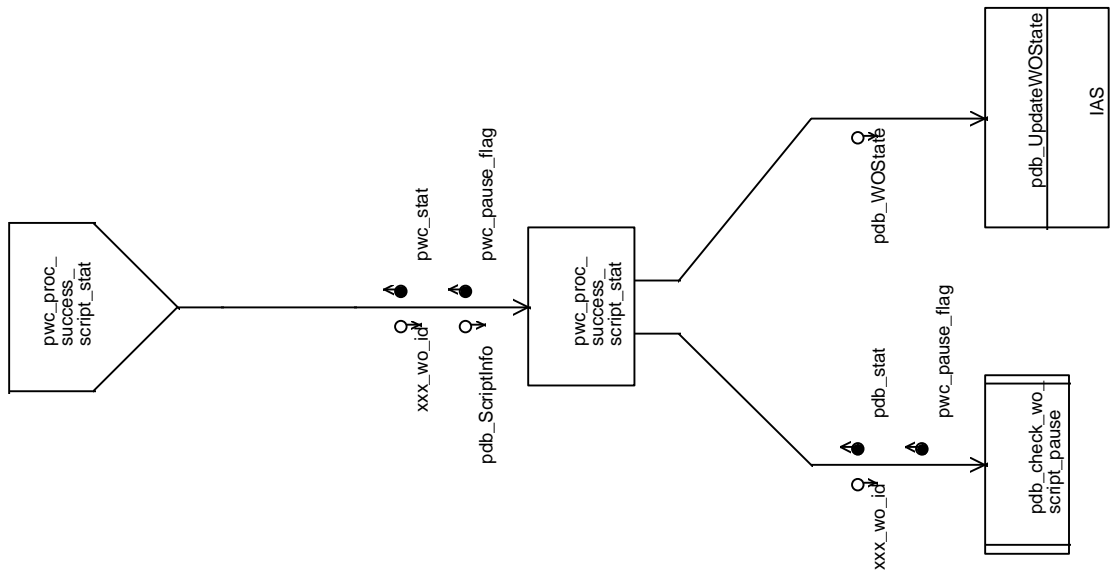
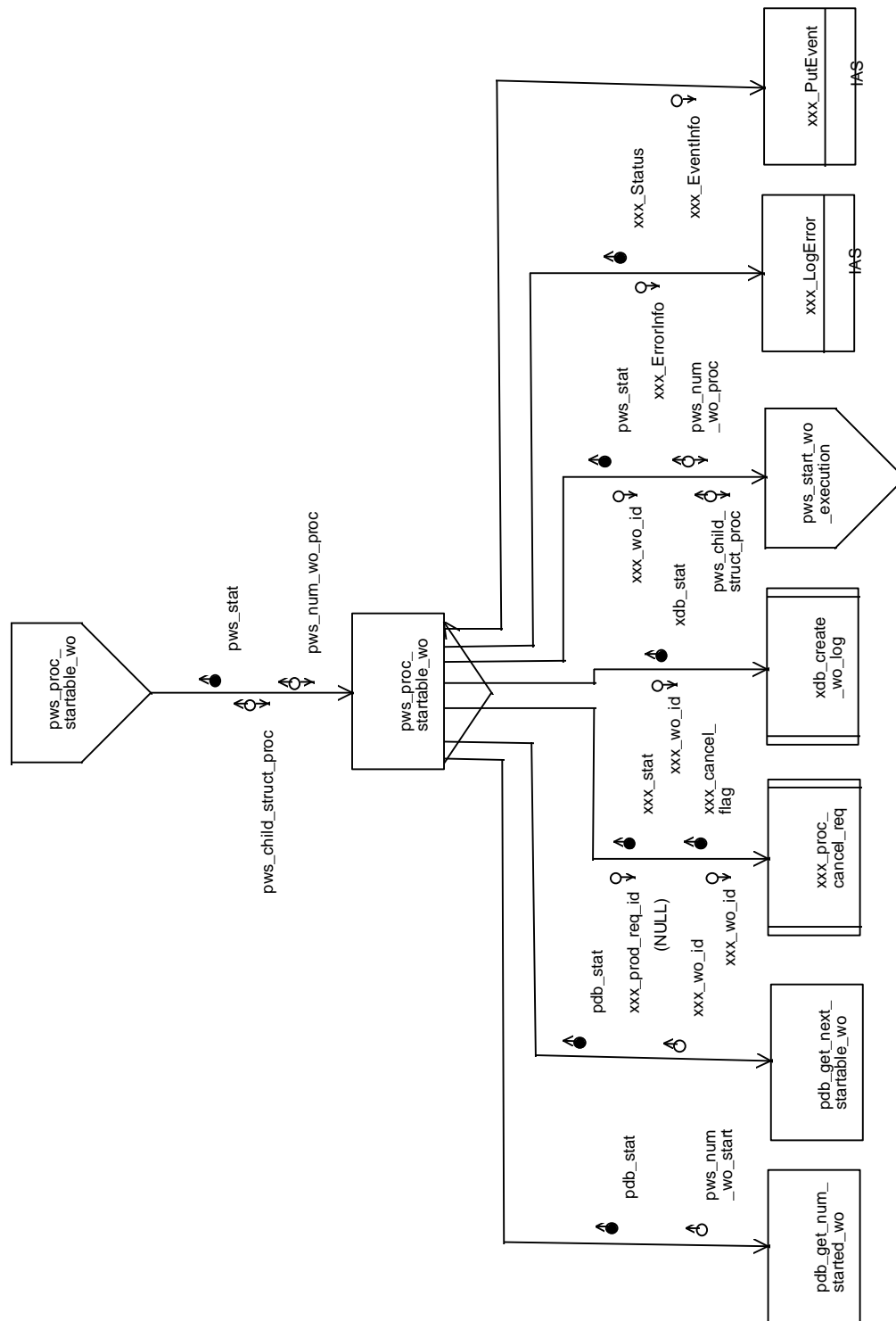


Figure 5-4. PWS Structure Chart (3 of 7)

## REVIEW



**Figure 5-4. PWS Structure Chart (4 of 7)**

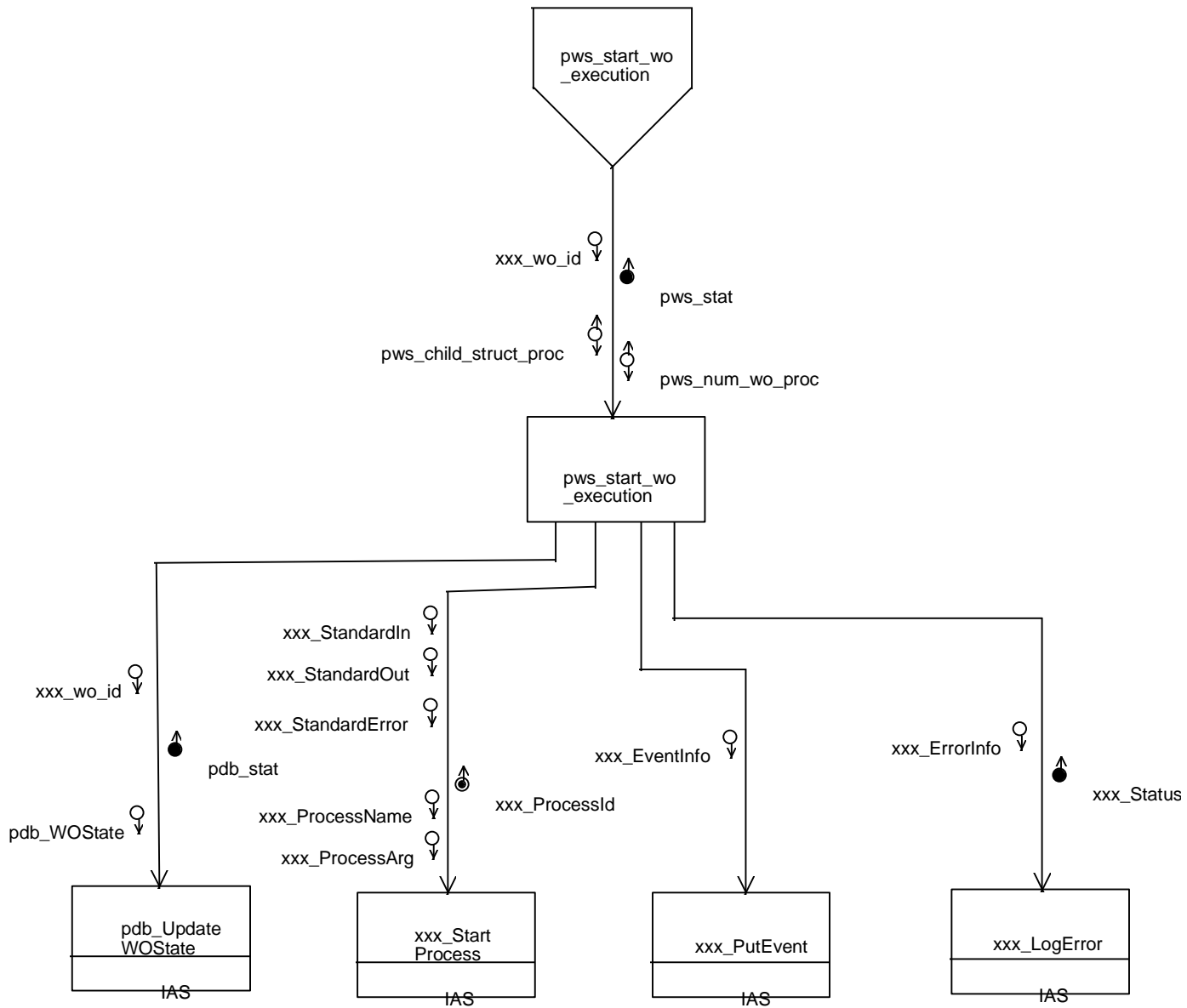


Figure 5-4. PWS Structure Chart (5 of 7)

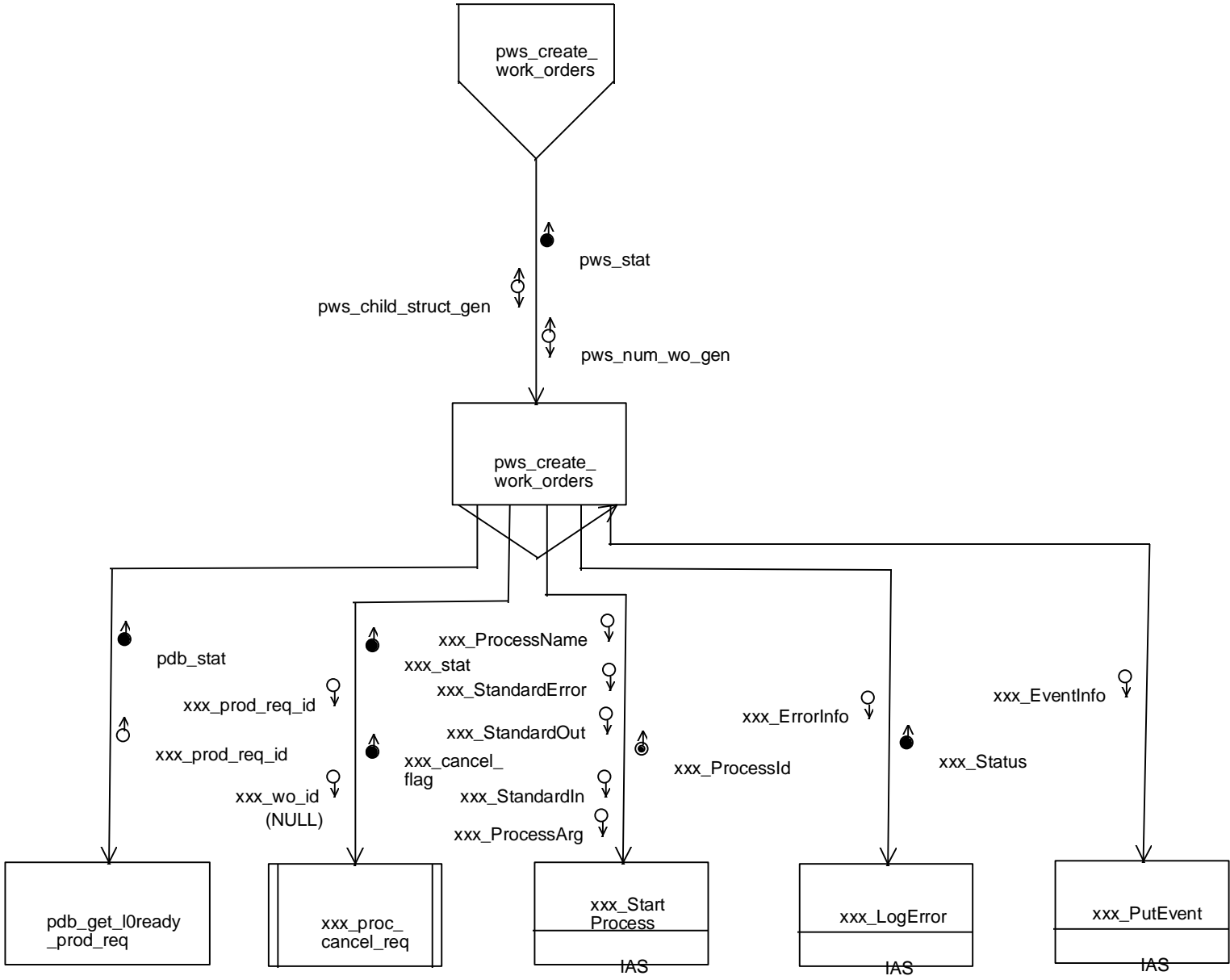


Figure 5-4. PWS Structure Chart (6 of 7)

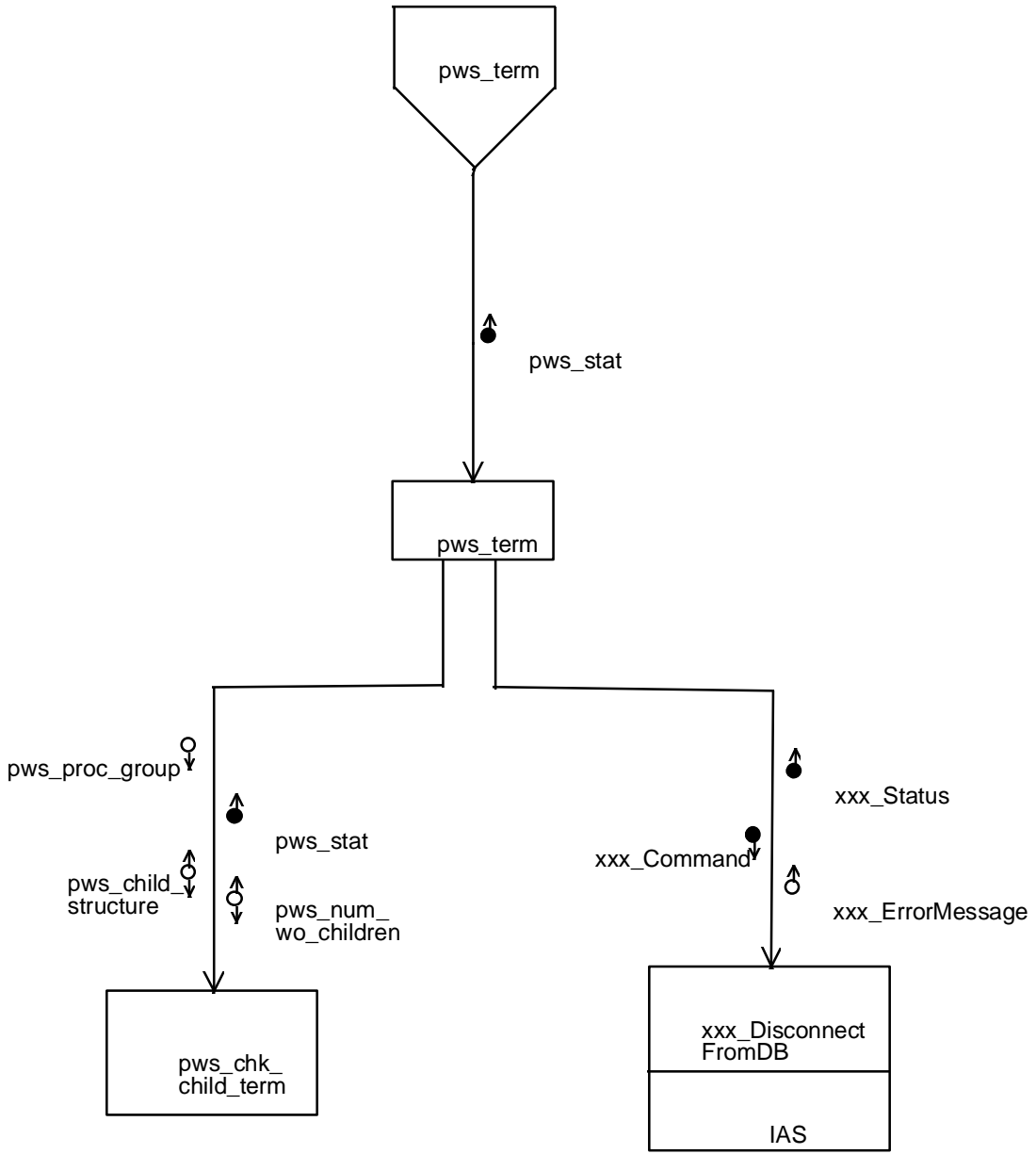


Figure 5-4. PWS Structure Chart (7 of 7)

## REVIEW

### **pdb\_get\_next\_startable\_wo—Get Next Startable Work Order**

#### **Parameters:**

pdb\_stat : control\_out  
xxx\_wo\_id : data\_out

**Body:** This module gets xxx\_wo\_id, the identification of the next work order whose processing can be started. Normally this will be the earliest work order in the pending state unless another work order in the same state has been promoted to take its place.

Errors encountered in the module are returned in pdb\_stat.

### **pdb\_get\_num\_started\_wo—Get Number of Started Work Orders**

#### **Parameters:**

pdb\_stat : control\_out  
pws\_num\_wo\_start : data\_out

**Body:** This module determines pws\_num\_wo\_start, the number of work orders whose processing has started or whose processing has completed but whose files have not yet been deleted from the LPGS, by querying the Work Orders and Product Requests tables in the database. The number is used as a measure of LPGS disk space usage to limit the processing of new work orders.

Errors encountered in the module are returned in pdb\_stat.

### **pws\_chk\_child\_term—Check Child Termination**

#### **Parameters:**

pws\_child\_structure : data\_inout  
pws\_proc\_group : data\_in  
pws\_num\_wo\_children : data\_inout  
pws\_stat : control\_out

**Body:** This module determines based on pws\_proc\_group whether any work order control processes or work order generation processes forked by the PWS task have terminated. If so, it decrements pws\_num\_wo\_children, the total number of such processes that have terminated since the last check was made and deletes those processes from the pws\_child\_structure.

Errors encountered by the module are returned in pws\_stat.

#### **PDL:**

DO UNTIL no more child termination requests are detected  
USE waitpid and wstat to check whether a child in the pws\_proc\_group has

## REVIEW

```
        terminated
    IF a child has terminated
        DECREMENT pws_num_wo_children
        REMOVE the child's process id from the pws_child_structure
    ENDIF
    IF a fatal error is encountered in the loop THEN
        CALL xxx_PutEvent to store the error in the Event table of the LPGS database
        CALL xxx_LogError to write the error to the PCS log file
        RETURN pws_stat with error
    ENDIF
ENDDO
RETURN pws_stat with success, pws_num_wo_children, pws_child_structure
```

### **pws\_create\_work\_orders—Create Work Orders**

#### **Parameters:**

pws\_stat : control\_out  
pws\_child\_struct\_gen : data\_inout  
pws\_num\_wo\_gen : data\_inout

**Body:** This module is responsible for generating new work orders for product requests whose L0 data is already available on LPGS for processing. The module loops through all such product requests until either pws\_num\_wo\_gen exceeds a prespecified threshold or no more l0ready product requests are found.

Within the loop, the module first fetches the next ready product request. If a suitable product request was found, the module first makes sure no cancel request is pending against the product request. If none is, it then starts an instance of the PWG task as an independent child process to create a new work order, updates the pws\_child\_struct\_gen structure with the process id of the new child process, and increments pws\_num\_wo\_gen, the actual number of work order creation processes running.

Processing errors encountered by the module are returned in pws\_stat.

#### **PDL:**

```
DO WHILE more product requests exist for which work orders can be created (state =
    L0READY) and pws_num_wo_gen does not exceed its threshold
    CALL pdb_get_l0ready_prod_req to get xxx_prod_req_id for the next product
        request for which L0 data is available
    IF an L0-ready product request was found THEN
        CALL xxx_proc_cancel_req to check whether a cancel request exists against the
            product request, perform the cancellation if it does, and return the
            xxx_cancel_flag to show the state of the cancellation
        IF the xxx_cancel_flag indicates that a cancellation did not need to be
            performed THEN
            CALL xxx_StartProcess to spawn a child process (PWG) to generate a work
```

## REVIEW

```
        order
        ADD the process id of the new child process to the pws_child_struct_gen
        structure
        INCREMENT pws_num_wo_gen, the number of work order generation
        processes running
    ENDIF
ENDIF
IF an error is encountered within an xxx or pdb routine in the loop THEN
    CALL xxx_LogError to write the error message to the LPGS log
    IF the error is fatal THEN
        CALL xxx_PutEvent to record the error message in the Event table
        RETURN pws_stat with error
    ELSE
        IF the error has occurred more than a prespecified number of times THEN
            RETURN pws_stat with error
        ELSE
            PROCEED to the next cycle
        ENDIF
    ENDIF
ENDIF
ENDDO
RETURN pws_stat with success, pws_child_struct_gen, pws_num_wo_gen
```

### **pws\_init—PWS Initialization**

#### **Parameters:**

pws\_stat : control\_out

**Body:** This module is responsible for initializing the PWS task. Its main functions are to connect the task to the LPGS database, define signal handlers for catching signals received by the PWS task, retrieve configuration parameters to be used by the task, and preallocate structures to hold information about children processes still running. It returns pws\_stat to indicate whether it was successful or not.

#### **PDL:**

```
CALL xxx_ConnectToDB to connect to the database
IF an error was encountered while connecting to the database THEN
    CALL xxx_LogError to write the error message to the LPGS log
    RETURN pws_stat with error
ENDIF
CALL xdb_get_task_config to get the PWS setup information from database
CALL xxx_Signal to define signal handlers for capturing the termination signals using
the appropriate handler names and process masks
USE Malloc to preallocate structures pws_child_struct_proc and pws_child_struct_gen
to hold lists of current work order controllers running and current work order
generators running, respectively
```

## REVIEW

```
IF a fatal error is encountered after connecting to the database THEN
    CALL xxx_LogError to write the error message to the LPGS log
    CALL xxx_PutEvent to write the error message to the event table in the database
    CALL xxx_DisconnectFromDB to disconnect from the database
    RETURN pws_stat with error
ENDIF
RETURN pws_stat with success
```

### **pws\_main—PWS Main Module**

**Parameters:** None

**Body:** This module first calls on pws\_init to initialize the PWS task. It then loops until it finds the xxx\_terminate flag set, in which case it invokes pws\_term to terminate the task.

While looping it first checks whether any Work Order controller child processes have terminated since the last check was made. It updates the number of such processes still running in the structure pws\_child\_struct\_proc based on the result.

The module then proceeds by calling pws\_proc\_resumable\_wo using pws\_child\_struct\_proc to resume the execution of any previously interrupted work orders and update the number of Work Order controller processes running. It then calls pws\_proc\_startable\_wo using pws\_child\_struct\_proc to process any new work orders that can be started and again update the number of Work Order controller processes running.

Finally, it checks whether any Work Order generation child processes have terminated since the last check was made and updates the number of such processes still running in the structure pws\_child\_struct\_gen based on the result. It then calls pws\_create\_work\_orders using pws\_child\_struct\_gen to start child processes that generate work orders before restarting the loop.

#### **PDL:**

```
CALL pws_init to initialize the current task
DO UNTIL xxx_terminate flag indicates that a SIGTERM signal has been received
    CALL pws_chk_child_term to check for PWC child termination and update
        pws_child_struct_proc by the number of terminated WO controller child
        processes
    CALL pws_proc_resumable_wo to process resumable work orders and update
        pws_child_struct_proc by the number of resumed WO controller child processes
    CALL pws_proc_startable_wo to process new work orders that can be started and
        update pws_child_struct_proc by the number of started WO controller child
        processes
    CALL pws_chk_child_term to check for pwg child termination and update
        pws_child_struct_gen by the number of terminated WO generation child
        processes
    CALL pws_create_work_orders to create new work orders and update
        pws_child_struct_gen by the number of started WO generation child processes
    SLEEP for a prespecified time interval (pws_interval)
```

## REVIEW

```
IF an error has occurred within the loop THEN
    IF the error is fatal or has occurred more than a prespecified number of
        times THEN
        DO FOR each running WO controller child process
            CALL kill to send a SIGTERM signal to the child process
        ENDDO
        CALL xxx_DisconnectFromDB to disconnect from the database
        EXIT the task
    ELSE
        PROCEED to the next cycle in the loop
    ENDIF
ENDIF
ENDDO
CALL pws_term to shut-down the current task gradual
EXIT the task
```

### **pws\_proc\_resumable\_wo—Process Resumable Work Orders**

#### **Parameters:**

pws\_num\_wo\_proc : data\_inout  
pws\_stat : control\_out  
pws\_child\_struct\_proc : data\_inout

**Body:** This module is responsible for continuing the processing of interrupted work orders whose resumption has been approved by the AAS analyst or operator. The module loops through all such work orders until either pws\_num\_wo\_proc exceeds a prespecified threshold or no more resumable work orders are found.

Within the loop, the module first fetches the next resumable work order. If a work order is found, the module first makes sure no cancel request is pending against the product request with which it is associated. If none is, it then starts an instance of the PWC task as an independent child process to process the work order, updates the pws\_child\_struct\_proc structure with the process id of the new child process, and increments pws\_num\_wo\_proc, the actual number of work order control processes.

Processing errors encountered by the module are returned in pws\_stat.

#### **PDL:**

```
DO UNTIL no suitable WO is found or pws_num_wo_proc exceeds its threshold
    CALL pdb_get_next_resumable_wo to get xxx_wo_id for the next WO whose
        execution can be resumed
    IF a resumable WO was found THEN
        CALL xxx_proc_cancel_req to check whether a cancel request exists against the
            current WO's product request, perform the cancellation if it does, and
            return the xxx_cancel_flag to show the state of the cancellation
        IF the xxx_cancel_flag indicates that a cancellation did not need to be
```

## REVIEW

```
    performed THEN
    CALL pws_start_wo_execution to start execution of the work order
        specified by xxx_wo_id using pws_child_struct_proc and
        pws_num_wo_proc
    IF a fatal error was encountered THEN
        RETURN pws_stat with error
    ENDIF
ENDIF
ENDIF
IF an error is encountered within an xxx or pdb routine in the loop THEN
    CALL xxx_LogError to write the error message to the LPGS log
    IF the error is fatal THEN
        CALL xxx_PutEvent to record the error message in the Event table
        RETURN pws_stat with error
    ELSE
        IF the error has occurred more than a prespecified number of times THEN
            RETURN pws_stat with error
        ELSE
            PROCEED to the next cycle
        ENDIF
    ENDIF
ENDIF
ENDDO
RETURN pws_stat with success, pws_child_struct_proc, pws_num_wo_proc
```

### **pws\_proc\_startable\_wo—Process Startable Work Orders**

#### **Parameters:**

pws\_child\_struct\_proc : data\_inout

pws\_stat : control\_out

pws\_num\_wo\_proc : data\_inout

**Body:** This module is responsible for starting the processing of new work orders for which L0 data is available. The module loops through all such work orders until either pws\_num\_wo\_proc exceeds a prespecified threshold, pws\_num\_wo\_start, the number of started or completed work orders whose files still reside on LPGS exceeds another threshold, or no more startable work orders are found.

Within the loop, the module first fetches the next startable work order. If a work order is found, the module makes sure no cancel request is pending against the product request with which it is associated. If none is, the module creates a work order log and starts an instance of the PWC task as an independent child process to process the work order, updates the pws\_child\_struct\_proc structure with the process id of the new child, and increments pws\_num\_wo\_proc, the actual number of work order control processes running.

## REVIEW

Processing errors encountered by the module are returned in pws\_stat.

### PDL:

```
DO UNTIL no suitable WO is found or pws_num_wo_proc exceeds its threshold or
pws_num_wo_start exceeds its threshold
CALL pdb_get_num_started_wo to get pws_num_wo_start, the number of started
or completed work orders whose files are still on LPGS
IF pws_num_wo_start will support the processing of an additional WO THEN
CALL pdb_get_next_startable_wo to get the next PENDING WO from the
Work Orders table
IF a startable WO was found THEN
CALL xxx_proc_cancel_req to check whether a cancel request exists against
the current WO's product request, perform the cancellation if it does, and
return the xxx_cancel_flag to show the state of the cancellation
IF the xxx_cancel_flag indicates that a cancellation did not need to be
performed THEN
CALL xdb_create_wo_log using xxx_wo_id to create a work order log
file in the work order's temporary directory
CALL pws_start_wo_execution to start execution of the work order
specified by xxx_wo_id using pws_child_struct_proc and
pws_num_wo_proc
IF a fatal error was encountered THEN
RETURN pws_stat with error
ENDIF
ENDIF
ENDIF
ENDIF
IF an error is encountered within an xxx, xdb or pdb routine in the loop THEN
CALL xxx_LogError to write the error message to the LPGS log
IF the error is fatal THEN
CALL xxx_PutEvent to record the error message in the Event table
RETURN pws_stat with error
ELSE
IF the error has occurred more than a prespecified number of times THEN
RETURN pws_stat with error
ELSE
PROCEED to the next cycle
ENDIF
ENDIF
ENDIF
ENDDO
RETURN pws_stat with success, pws_child_struct_proc, pws_num_wo_proc
```

## REVIEW

### **pws\_start\_wo\_execution—Start Work Order Execution**

#### **Parameters:**

pws\_stat : control\_out  
xxx\_wo\_id : data\_in  
pws\_hild\_struct\_proc : data\_inout  
pws\_num\_wo\_proc : data\_inout

**Body:** This module is responsible for starting a child process to control script execution for new or interrupted work orders. When the child has been started, the module increments pws\_num\_wo\_proc and updates pws\_child\_struct\_proc. Processing errors encountered by the module are returned in pws\_stat.

#### **PDL:**

```
CALL pdb_UpdateWOState to set the WO state in the Work Orders table to
EXECUTING
CALL xxx_StartProcess to spawn a child process (PWC) to control further
processing of the WO using xxx_wo_id
INCREMENT pws_num_wo_proc, the number of WO control processes still running
ADD the process id of the new child process to pws_child_struct_proc
IF an error is encountered in any of the above submodules THEN
    CALL xxx_LogError to write the error message to the LPGS log
    IF the error is fatal THEN
        CALL xxx_PutEvent to store the error in the Event table
    ENDIF
    RETURN pws_stat with error
ENDIF
CALL xxx_PutEvent to store the event in the Event table
RETURN pws_stat with success, pws_child_struct_proc, pws_num_wo_proc
```

### **pws\_term—PWS Gradual Termination**

#### **Parameters:**

pws\_stat : control\_out

**Body:** This module is responsible for terminating the PWS task when the task receives a SIGTERM signal. Its main functions are to check which controller (PWC) children are still running, send a gradual termination message to those children, and wait for all its running children (PWC and PWG) to complete. The module then disconnects the PWS task from the LPGS database. Errors encountered during termination are returned in pws\_stat.

#### **PDL:**

```
CALL pws_chk_child_term to check for terminated WO controller child processes and
update pws_child_struct_proc by the number of terminated WO controller child
processes
```

## REVIEW

```
DO FOR each running WO controller (PWC) child process
    CALL kill to send a SIGTERM (15) signal to the current PWC child
ENDDO
DO UNTIL pws_num_wo_gen indicates that all pwg children have terminated and
    pws_num_wo_proc indicates that all PWC children have terminated
    CALL pws_chk_child_term to check for pwg child termination and update
        pws_child_struct_gen by the number of terminated WO generation child
        processes
    CALL pws_chk_child_term to check for PWC child termination and update
        pws_child_struct_proc by the number of terminated WO controller child
        processes
    SLEEP for a prespecified time
    IF an error has occurred within the loop THEN
        IF the error is fatal or has occurred more than a prespecified number of
            times THEN
            EXIT the loop
        ELSE
            PROCEED to the next cycle in the loop
        ENDIF
    ENDIF
ENDDO
CALL xxx_DisconnectFromDB to disconnect from the database
RETURN pws_stat
```

### 5.3.4 Work Order Controller (PWC) Task

This subsection describes the PWC task software.

#### 5.3.4.1 Task Overview

The Work Order Controller task is responsible for starting and monitoring the work order scripts for the work order specified as an input parameter supplied by the PWS task. It continues processing scripts until the last script completes successfully, an error occurs, or a halt is encountered. Before starting a new script it checks to see if the work order needs to be canceled. When there are no more scripts to process and all QA checks have passed, PWC updates the product request state to indicate that the product is ready for transfer to ECS. If an error occurs that requires investigation, PWC notifies the AAS analyst by inserting an entry in the event log and recording the failure in the work order table.

#### 5.3.4.2 Initialization

The PWC task initialization consists of connecting to the database, setting up the signal catcher for the termination message, and retrieving the work order identifier supplied as an input parameter to the task.

## REVIEW

### 5.3.4.3 Normal Operation

PWC begins processing by calling `pwc_init` to initialize the task. Then it performs a loop until it completes processing of the last script, encounters a halt, or encounters a failure. Before starting a script, it calls `xxx_proc_cancel_req` to process the cancellation if the product request needs to be canceled. If the request has not been canceled, PWC calls `pwc_StartNextScript` to run the next script. Then it calls `pwc_MonitorScript` to wait for completion of the script. When the script completes, it returns a status indicating success or failure. If the script failed, PWC calls `pdb_Update_WOState` to update to work order state to indicate the error. Otherwise, PWC calls `pwc_proc_success_script_stat` to process the successful completion and to check whether the work order is to be paused. If the last script for the work order completed successfully and there is no pause, PWC calls `pwc_proc_completed_wo` to complete the work order processing phase. Part of this processing is updating the product request state (for requests that need to be shipped) to indicate to the DMS that the product can be shipped. When there are no more scripts or a halt is encountered, `xxxDisconnectFromDB` is called to disconnect the task from the database prior to termination.

### 5.3.4.4 Error Handling

The Work Order Controller reports any errors encountered to the system log file and/or the event log and exits gracefully when it is unable to continue processing.

### 5.3.4.5 Design

This subsection presents the design of the PWC task. Figure 5-5 shows the structure chart for the Work Order Controller task. The module specifications for the PWC task are provided below.

#### **`pdb_check_wo_script_pause`—Check Work Order Script Pause**

##### **Parameters:**

`pwc_pause_flag` : control\_out

`xxx_wo_id` : data\_in

`pdb_stat` : control\_out

**Body:** This module checks to see if there is a pause set for work order `xxx_wo_id` after the script that just completed. The module sets the `pwc_pause_flag` to TRUE if a pause has been found and return `pdb_stat` to indicate successful or unsuccessful completion.

REVIEW

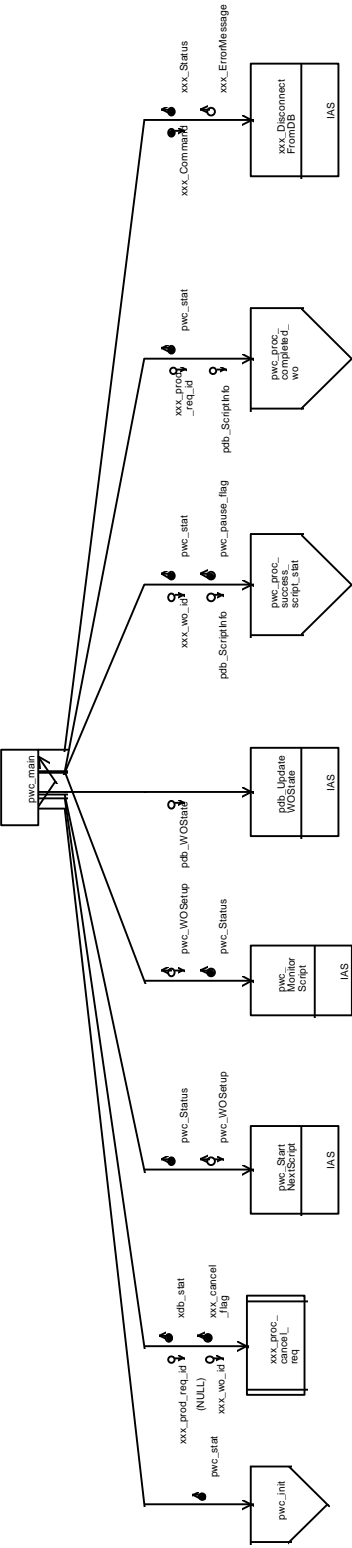
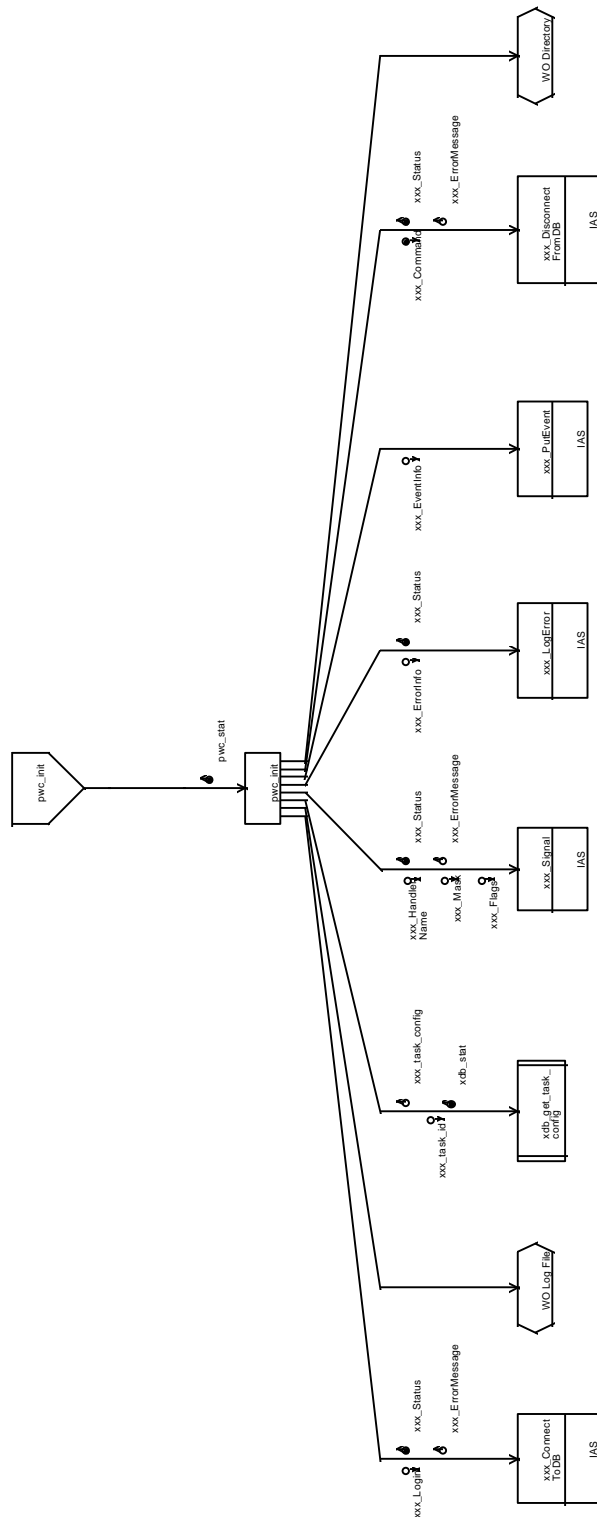


Figure 5-5. PWC Structure Chart (1 of 4)

## REVIEW



**Figure 5-5. PWC Structure Chart (2 of 4)**

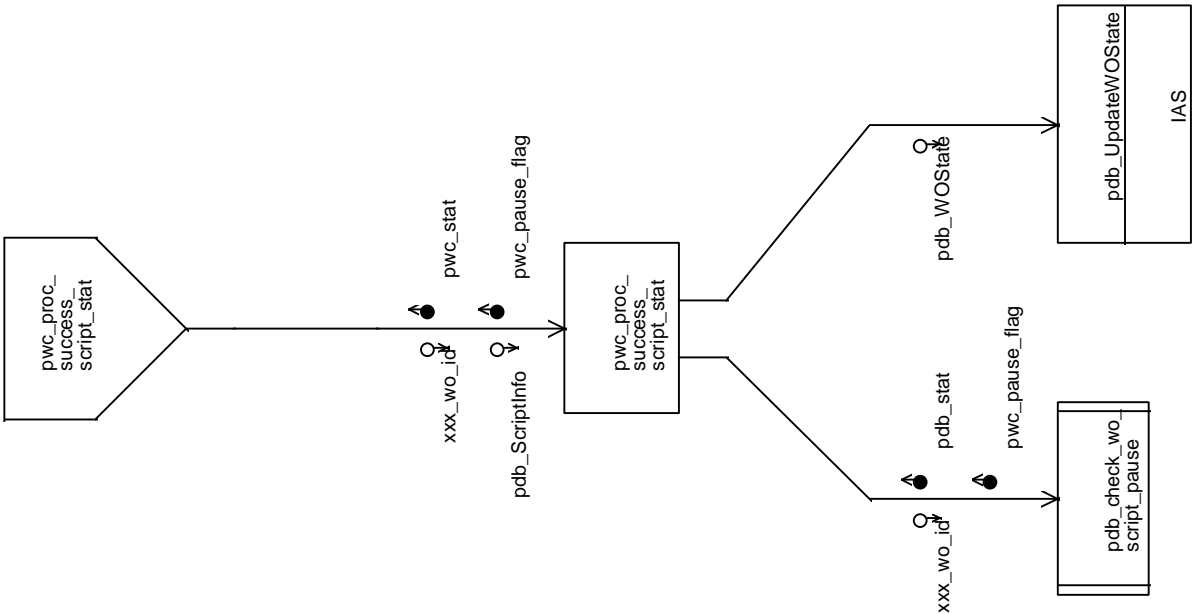


Figure 5-5. PWC Structure Chart (3 of 4)

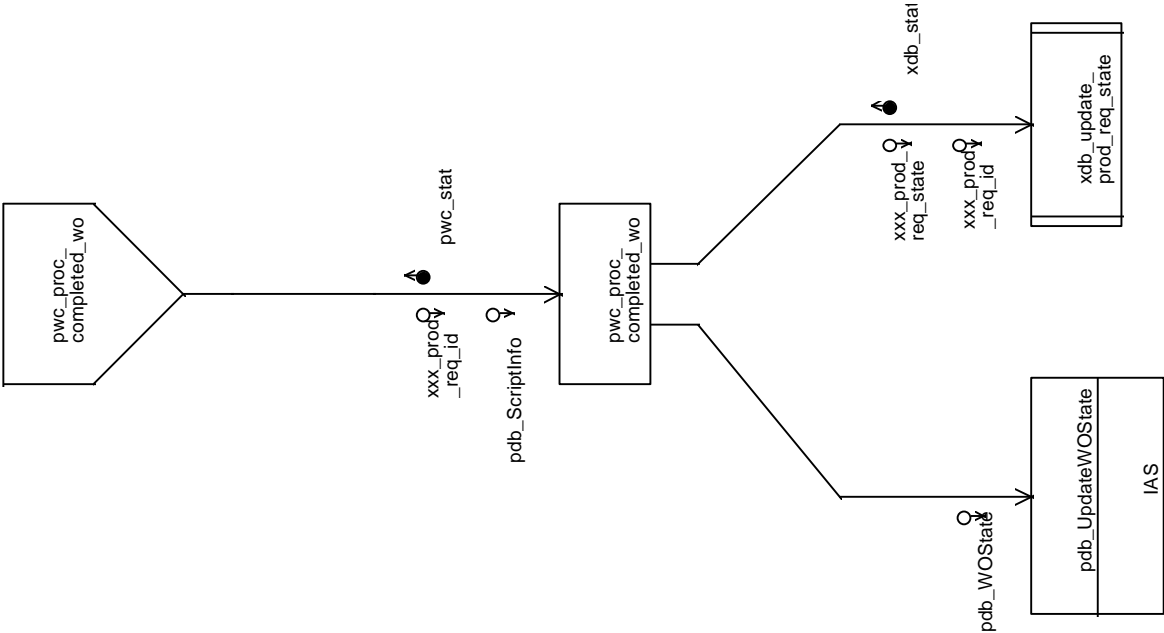


Figure 5-5. PWC Structure Chart (4 of 4)

## REVIEW

### **pwc\_init—PWC Initialization**

**Parameters:** None

**Body:** This module initializes the Work Order Controller task. It connects to the database; sets up environmental variables based on the configuration file; sets up signal handlers; and updates the log file. If it encounters problems, it disconnects from the database and writes the error to the error log.

### **pwc\_main—Control Work Order Execution**

**Parameters:** None

**Body:** This module is invoked when there are more scripts to execute for a work order identified by xxx\_wo\_id. pwc\_init is first called to initialize the task. A set of commands is then executed in a loop until either there are no more scripts for the work order, a halt condition has been encountered, the script failed, or the work order has completed.

The loop starts with a loop that looks for a gradual shutdown signal. Then the loop calls xxx\_proc\_cancel\_req to see if the product request associated with the xxx\_wo\_id has been canceled. If the product request has been canceled this module processes the cancellation.

If the product request has not been canceled, this module calls the pwc\_StartNextScript module to determine which is the next script to run for this work order as well as to start the script.

The pwc\_MonitorScript module is then invoked to wait for the script to terminate. When the script terminates, the wstat variable is checked. If wstat indicates a problem, the pdb\_UpdateWOState module is invoked to update the work order state to ANOMALY. If wstat indicates that the script succeeded, this module invokes the pwc\_proc\_success\_script\_stat to check if there is a halt associated with the script. If there is a halt associated with the script then the work order state is updated to HALTED, the xxx\_DisconnectFromDB module is invoked to disconnect from the database, and the task is terminated.

If pwc\_StartNextScript indicates that the work order has completed then this module calls pwc\_proc\_completed\_wo to update the work order state to READY TO SHIP and the product request state to OUT READY. The xxx\_DisconnectFromDB module is invoked to disconnect from the database and the task is terminated.

#### **PDL:**

```
SET exit_loop to FALSE
CALL pwc_init to initialize data, get database access, and to set up a signal
    catcher
IF pwc_init is not successful THEN
    Return with error status
DO UNTIL a gradual shut-down signal is received
    DO WHILE exit_loop is set to FALSE
        IF gradual shutdown flag is set THEN
            EXIT loop
```

## REVIEW

```
END
CALL xxx_proc_cancel_req to check if the product request associated
    with the xxx_wo_id has been canceled and if so process the
    cancellation
IF product request has been canceled THEN
    SET exit_loop to TRUE
    EXIT loop
END IF
IF gradual shutdown flag is set THEN
    EXIT loop
END IF
CALL pwc_StartNextScript to get the next script that should
    run for this work order and start the script
IF pwc_StartNextScript returns a status indicating that the work
    order has completed THEN
    CALL pwc_proc_completed_wo to update necessary states
    SET exit_loop to TRUE
    EXIT loop
END IF
CALL pwc_MonitorScript to wait for the script to end and
    to process the exit status
IF script status indicates that the script failed THEN
    CALL pdb_UpdateWOState to update the work order state
        to anomaly
    SET exit_loop to TRUE
ELSE
    CALL pwc_proc_success_script_stat to check if a halt is
        associated with this script and if so update the
        work order state to indicate the halt
    IF script has halt THEN
        SET exit_loop to TRUE
    END IF
END IF
END WHILE
END DO
CALL xxx_DisconnectFromDB to terminate the task
```

### **pwc\_proc\_completed\_wo—Process Completed Work Order**

#### **Parameters:**

xxx\_prod\_req\_id : data\_in  
pwc\_stat : control\_out  
pdb\_ScriptInfo : data\_in

## REVIEW

**Body:** This module is invoked when there are no more scripts to run for the work order. It calls pdb\_UpdateWOState module to update the work order state to READY TO SHIP. If the work order type is not DIAGNOSTIC or BENCHMARK then this module calls xdb\_update\_prod\_req\_state to update the product request state to OUT READY. The module returns pwc\_stat to indicate successful or unsuccessful processing.

### **pwc\_proc\_success\_script\_stat—Process Successful Script Status**

#### **Parameters:**

pwc\_pause\_flag : control\_out

pwc\_stat : control\_out

pdb\_ScriptInfo : data\_in

**Body:** This module is invoked when a script has ended successfully. The pdb\_check\_wo\_script\_pause is invoked to check if the script has a pause associated with it. If there is a pause associated with the script then the pdb\_UpdateWOState module is invoked to set the state of the work order to halted. The pwc\_pause\_flag is set to TRUE and the pwc\_stat is set to indicate successful or unsuccessful completion.

# Section 6. Data Management Subsystem

---

## 6.1 Introduction

The DMS provides product generation request ingest, product ingest, product processing, product formatting, product transfer, resource monitoring, file deletion, and report generation. Also, the DMS is responsible for providing the interface with IAS and ECS.

## 6.2 Design Overview

This section provides an overview of the DMS software design. It presents the relationships between the DMS and the other LPGS subsystems. It also discusses the considerations and assumptions used in the design process.

### 6.2.1 Subsystem Software Overview

Figure 6-1 contains the DMS context diagram. As shown, the DMS interfaces with ECS, IAS, PCS, the LPGS operator, and the LPGS database. The DMS receives the product generation requests and the Level 0R (L0R) products from the ECS; the DMS notifies ECS when the Level 1 (L1) products are ready for delivery. In addition, the DMS sends and receives protocol messages that are associated with the exchange of these products with ECS. The DMS receives processing requests from the PCS; the DMS sends processing status to the PCS. The DMS also interfaces with the PCS indirectly via the LPGS database. The DMS is also responsible for periodically deleting the characterization statistics after IAS retrieves the data from the LPGS database. In addition, the DMS processes operator requests to delete product request files and directories, generate accounting reports, delete the characterization statistics, acknowledge L0R products ingested manually, and cancel the product generation request.

The DMS is responsible for portions of the nominal processing of the L1 product. The DMS receives the product generation requests from ECS and stores the requests in the LPGS database. Then, the DMS is responsible for ingesting the L0R product. The DMS ingest of the L0R product includes sending a request for the product to ECS, receiving a response from ECS when the product is available for transfer, retrieving the product via ftp, and sending ECS notification after the product is successfully transferred. After the ingest is completed, PCS notifies the DMS when the L0R product needs to be prepared for Level 1 product generation, and the DMS prepares the L0R products. After the L1 product generation is complete, the PCS notifies the DMS to convert the L1 product into the final format (HDF-EOS, GeoTIFF, or FAST-C). When the product is ready for transfer to ECS, the DMS is responsible for notifying ECS that the product is ready for retrieval. ECS retrieves the L1 product and notifies the DMS when the transfer is complete. Finally, the DMS periodically deletes the old products from the LPGS disk.

REVIEW

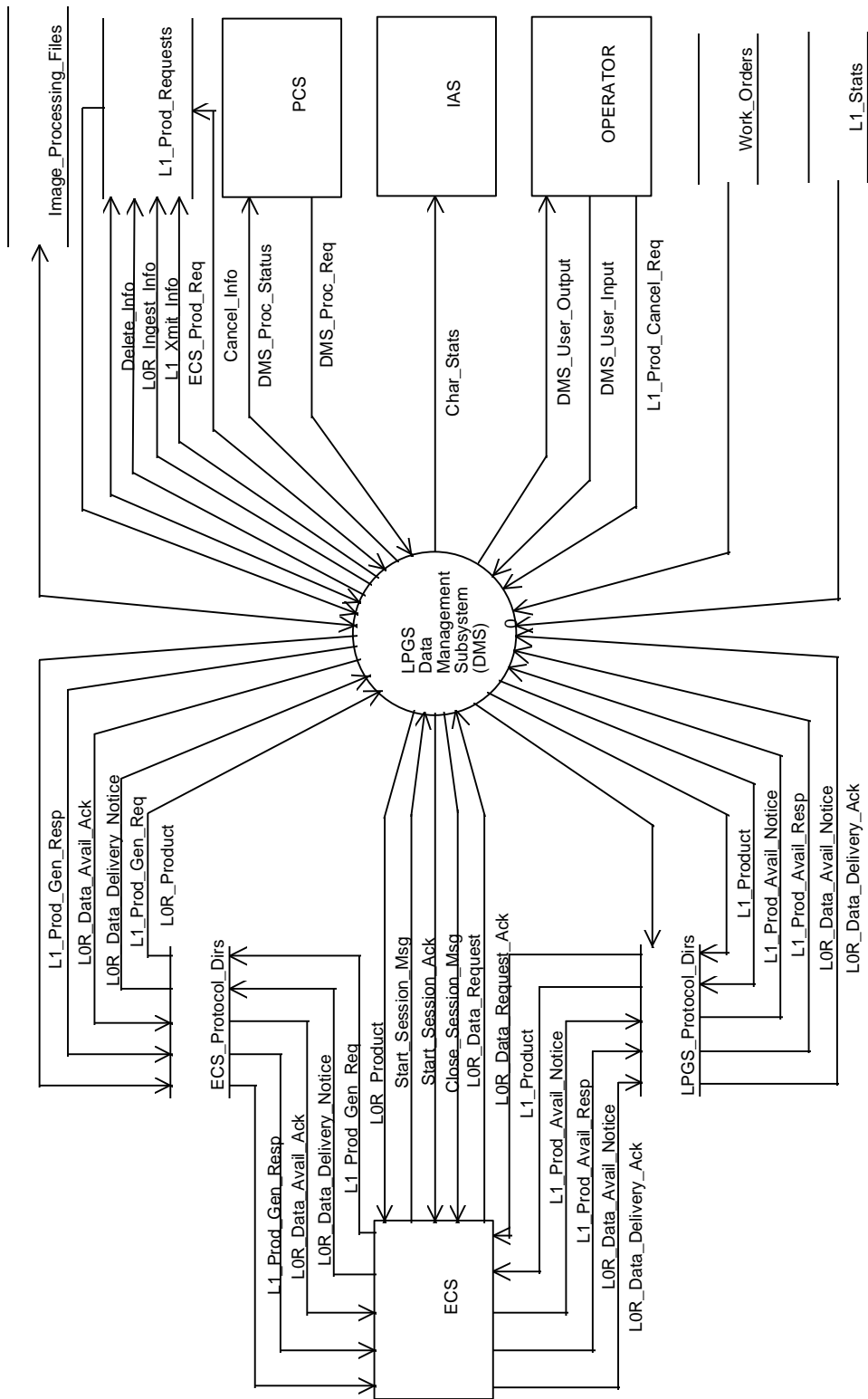


Figure 6-1. DMS Context Diagram

# REVIEW

## 6.2.2 Design Considerations/Assumptions

This subsection presents the design drivers relevant to the DMS software and the assumptions of the software design.

The following assumptions were made during the design of the DMS:

- The design for the DMS is based on the ECS/LPGS interface concepts working paper. The differences between this working paper and Interface Control Document Between EOSDIS Core System (ECS) and the ESDIS Level 1 Product Generation System (LPGS) only impacts the design of the DMS process that ingests the LOR products (DIL).
- COTS products will be used for
  - Converting the L1 Products to HDF-EOS format
  - Converting the L1G image to GeoTIFF and FAST-C format
  - FTPing files and POLLing directories on remote systems
  - Reading and writing files in ODL/PVL format
  - Accessing the database
- The DMS will reuse software for implementing the ECS interfaces.
- The DMS will reuse IAS software for the preprocessing of the LOR product and for the resource management task.
- The IAS software for preprocessing the LOR product includes software to generate the consensus PCD and MSCD.
- IAS retrieves the characterization statistics from the LPGS database, and updates the LPGS database to indicate the statistics were retrieved. After IAS has retrieved the statistics, the DMS deletes the statistics from the database.
- IAS can access the LPGS database via a network connection.
- Protocol files placed on the LPGS by ECS are deleted by the DMS tasks.
- Each product request will have a unique directory that contains multiple subdirectories including the input and save directories and a work order directory that is created each time the work order is run.
- The DMS ingest process will control when the LOR data is requested from ECS. Thresholds for the ingest process will be configurable by the operator.
- The final L1 product is retrieved via ftp by ECS from a single product request delivery directory.

## REVIEW

- The product request and work order files and directories are marked for deletion after the L1 product is successfully transferred to ECS.
- The DMS resource manager only deletes files and directories associated with the product request which have been marked for deletion.

### 6.3 Subsystem Design

This subsection provides a detailed description of the DMS software task model selected to implement the DMS design. The DMS software is designed as a single software configuration item (SWCI) that satisfies all the data management requirements imposed on the LPGS.

The transforms presented in the DMS essential model are grouped into seven independent tasks that execute concurrently. The DMS data flow diagram, shown in Figure 6-2, consists of the following tasks: IF with ECS (DIE), Ingest L0R Product (DIL), Process L0R Product (DPL), Format L1 Product (DFL), Xmit L1 Product (DXL), Resource Manager (DRM), and Generate Reports (DGR). The design of each of the DMS software task consists of a single software component.

The DIE task is responsible for interfacing with ECS for the receipt of the product generation requests. The task controls the transfers, which includes performing the handshaking protocol, ensuring the interface is available, and validating the product generation requests.

The DIL task is responsible for ingesting the L0R products. This task periodically sends ECS requests for L0R products. After the data is staged by ECS, this task retrieves the product via ftp. Then, this task stages the L0R data in the appropriate product request directory. Finally, DIL updates the database to indicate the L0R data is available for processing.

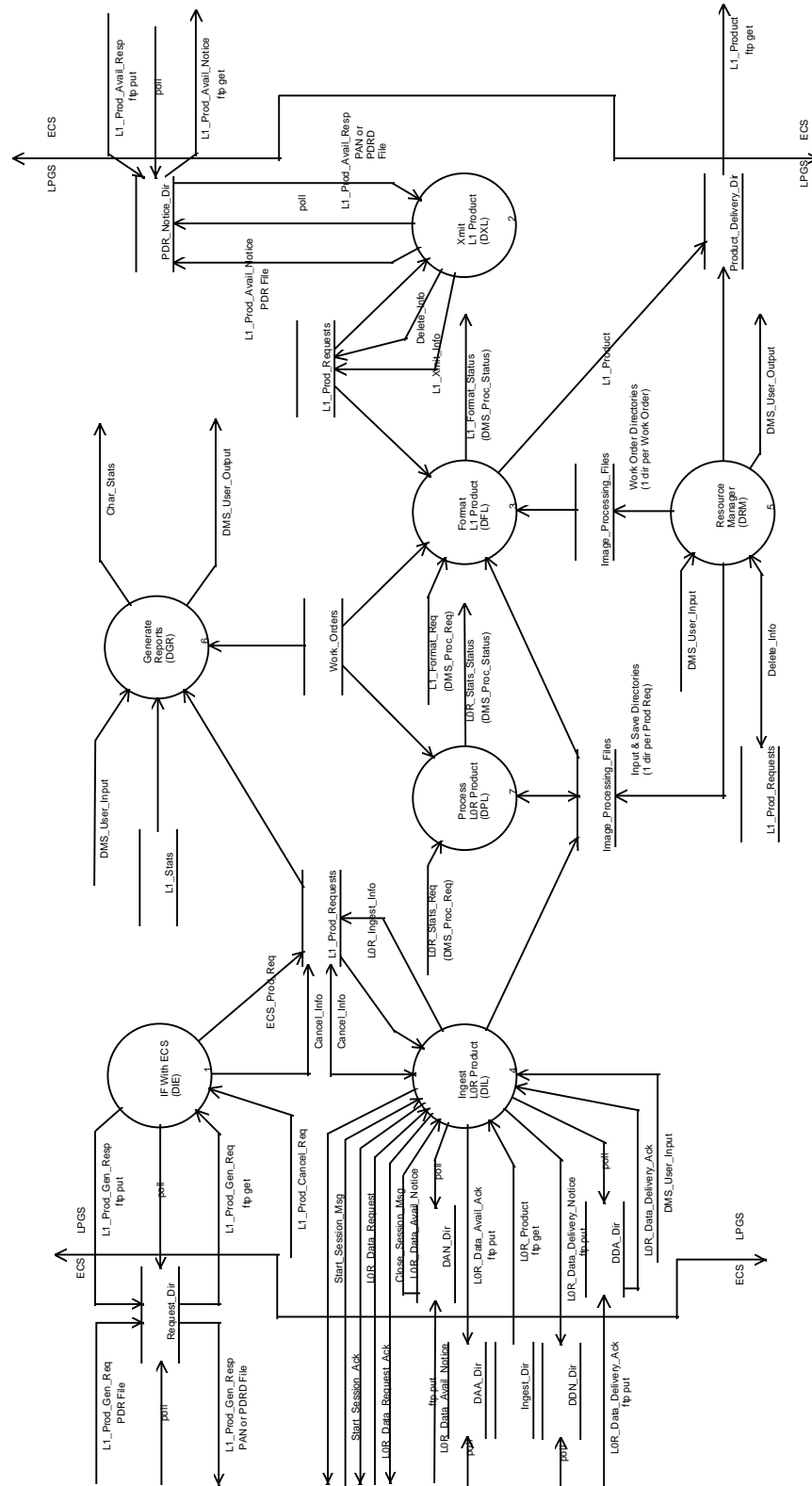
The DPL task prepares the L0R data for Level 1 product generation. This task ensures the data quality is appropriate for L1 product generation and generates characterization statistics. This task also generates the consensus PCD and MSCD files.

The DFL task converts the L1 product into the final format. This task converts the L1 output products into HDF-EOS format. Then, this task converts the L1G product into the specified format (GeoTIFF or FAST) if requested by the user. Finally, this task moves the product into the output directory and checks the product for completeness.

The DXL task is responsible for transmitting the L1 products. This task notifies ECS that the product is ready for transfer. After the product is retrieved by ECS via ftp, this task receives notification the product was transferred. Next, DXL updates the database to indicate the product was delivered and marks the files associated with the product request for deletion.

The DRM task is responsible for managing the data associated with the product requests and work orders. This task periodically deletes files and directories marked for deletion and reports the disk usage. Also, this task receives and processes operator requests to delete the product request and work order files and directories.

## REVIEW



**Figure 6-2. DMS Level 0 DFD**

## REVIEW

The DGR task is responsible for generating reports. This process deletes the characterization statistics after IAS retrieves the data from the LPGS database. Also, this task generates operator requested accounting reports.

The key objective used in the task-level design was to identify the primary services required of the DMS software and to evaluate their implementation as sequential or concurrent activities relative to each other. The seven tasks (DIE, DIL, DPL, DFL, DXL, DRM, and DGR) were selected as candidates for implementation as concurrent processes.

The global data stores that contain information used by tasks for performing the required functions and communicating with other tasks are shown in the task model. The DMS software design uses a local COTS DBMS to store and manipulate information about received image files, products, and reports generated from the received input files. The database contains setup and configuration information needed by the DMS to perform its required functions.

The description of each task is presented in the following subsections. This description consists of a task overview, initialization, normal operation, error handling, and design of the task.

### **6.3.1 DMS IF With ECS (DIE) Task**

This subsection describes the DIE task software.

#### **6.3.1.1 Task Overview**

The DIE task is responsible for retrieving the product generation User Request Files (URF) from a ECS controlled server. These URF files follow the ODL/PVL language structure and contain the information necessary for LPGS to generate the requested L1 product. The URF files are pulled from the ECS system using ftp. Then, DIE extracts the information from the URF and inserts a new record in the product request table in the database. Next, DIE sends ECS a URF acknowledgment file in response to the product generation request. The DIE task is also responsible for processing cancellation requests from the operator.

#### **6.3.1.2 Initialization**

This task is started by the PSI task and remains memory resident thereafter. The DMS configuration parameters are retrieved from the database.

#### **6.3.1.3 Normal Operation**

The DIE task periodically checks a predefined directory on the ECS controlled host for new URF files. DIE pulls (using ftp) any new files found and validates them. A response file, which provides the status of the request, is placed on the ECS host via ftp. If the URF passes validation, the information is extracted from the file and is inserted into the product request table in the LPGS database. If errors are encountered with the URF (except intersystem communication problems), the DIE task sends ECS an URF acknowledgment file which indicates the nature of the problem. When the DIE task receives a cancellation requests from the operator, a flag within the database is set to indicate the product request cancellation is pending.

## REVIEW

### 6.3.1.4 Error Handling

The DIE task establishes a signal handler to capture all fatal UNIX signals which would cause the task to abort. The signal handler sends an error message to the operator, disconnects from the database, and gracefully shuts down the task. Nonfatal errors are logged to the LPGS event log and an error notification is sent to the user interface.

### 6.3.1.5 Design

This subsection presents the design of the DIE task. The structure chart for the task is illustrated in Figure 6-3. The module specifications for the DIE task are provided (in alphabetical order) below.

#### **die\_create\_response—Build a Response File for a Received PDR**

##### **Parameters:**

die\_pdr\_filename : data\_in  
die\_resp\_filename : data\_inout  
die\_status : control\_out

**Body:** This module creates a response file based on the contents of die\_pdr\_filename. The filename is found in die\_resp\_filename. The ODL COTS software will be used to generate parameter names and values.

#### **die\_eval\_urf—Validate the Product Request Received from ECS**

##### **Parameters:**

die\_pdr\_fnlist : data\_inout  
die\_status : control\_out

**Body:** This module checks the contents of each Product Request File using the ODL COTS software. Valid Product Requests are inserted into the database using die\_format\_data. Status returned indicates whether the Product Request was OK or an indication of the problem encountered.

##### **PDL:**

```
DOFOR each file in die_pdr_fnlist
  READ parameters in file
  IF parameters valid THEN
    CALL die_format_data to enter data into database
    SET flag to indicate file valid
  ELSE
    SET flag to indicate error in parameters
  ENDIF
```

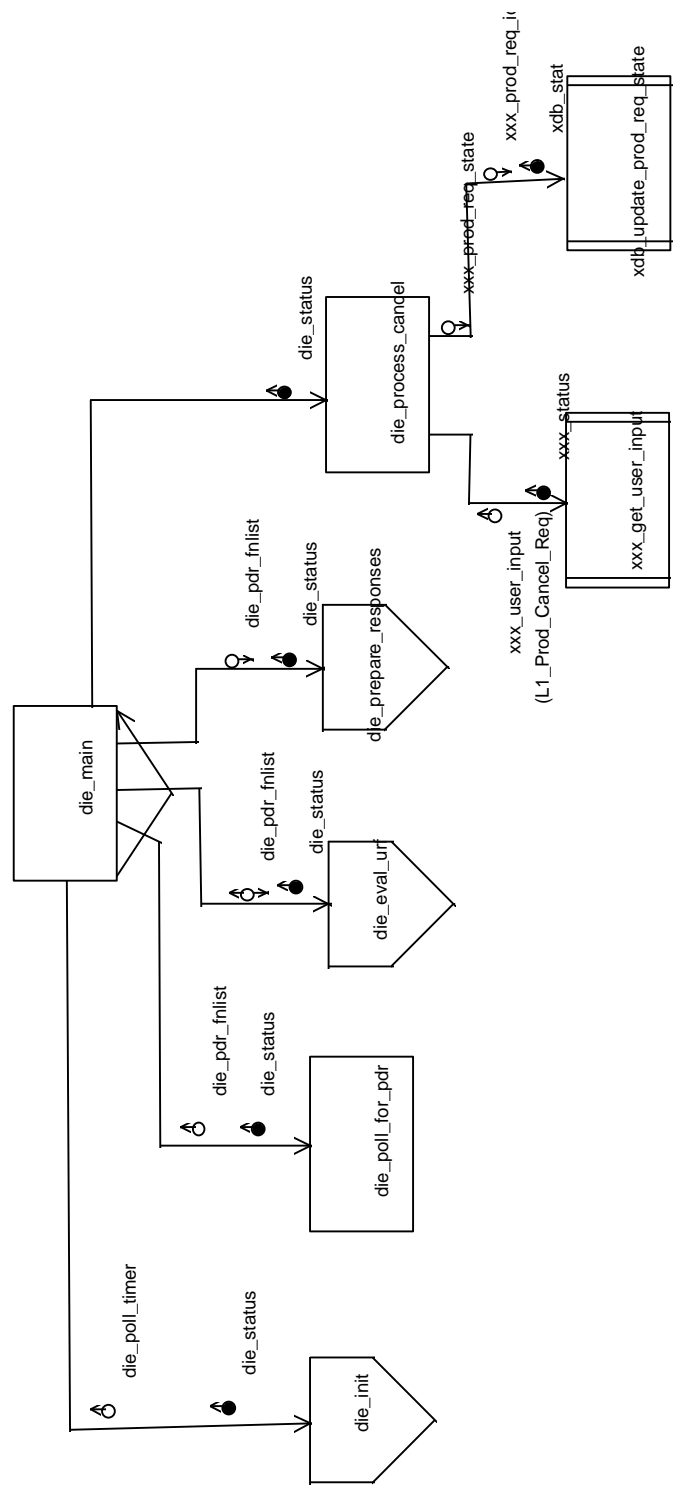


Figure 6-3. DIE Structure Chart (1 of 6)

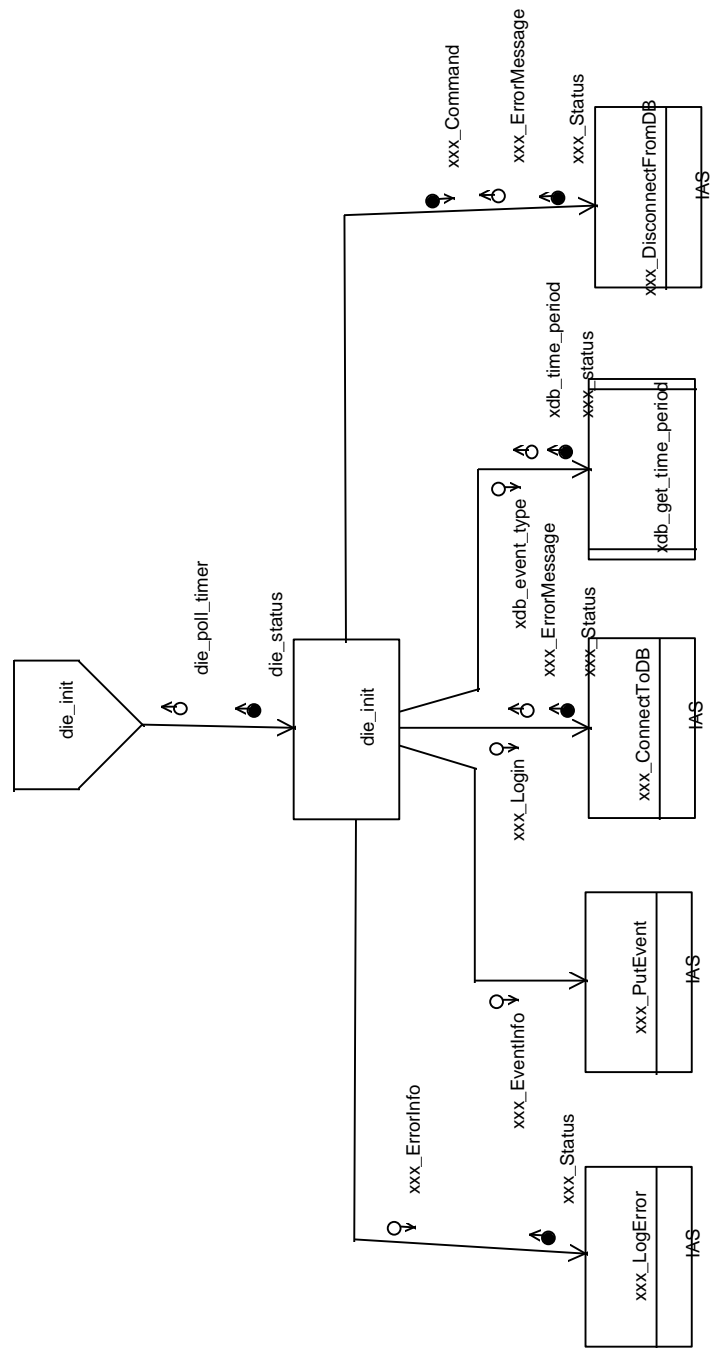


Figure 6-3. DIE Structure Chart (2 of 6)

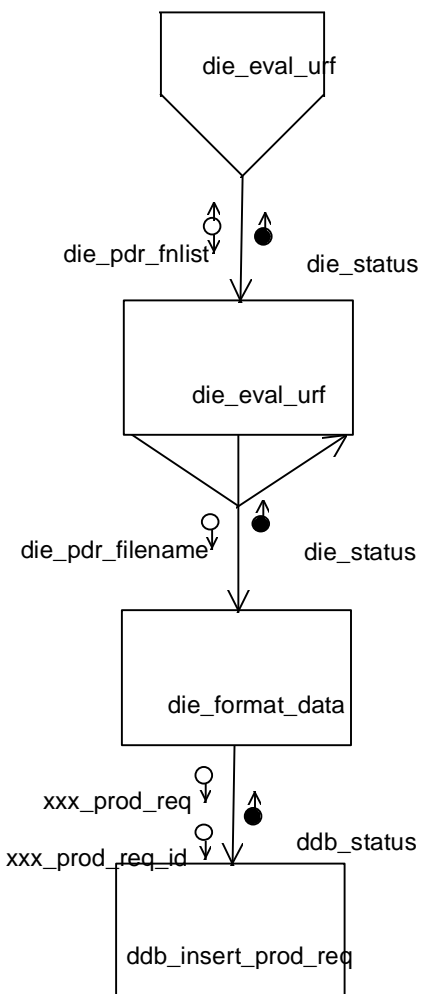


Figure 6-3. DIE Structure Chart (3 of 6)

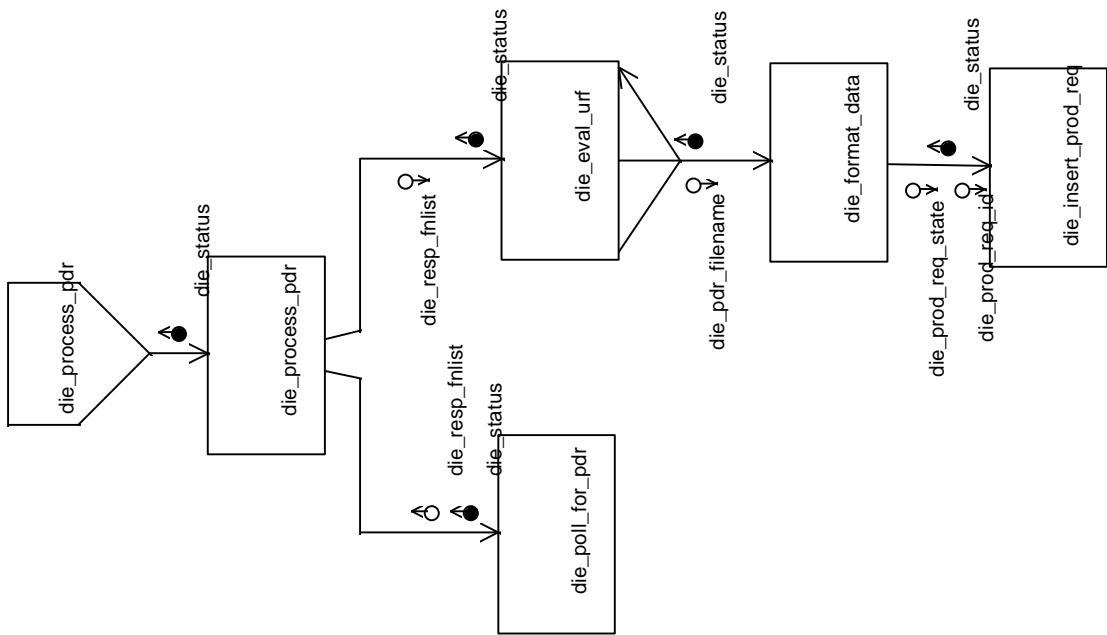


Figure 6-3. DIE Structure Chart (4 of 6)



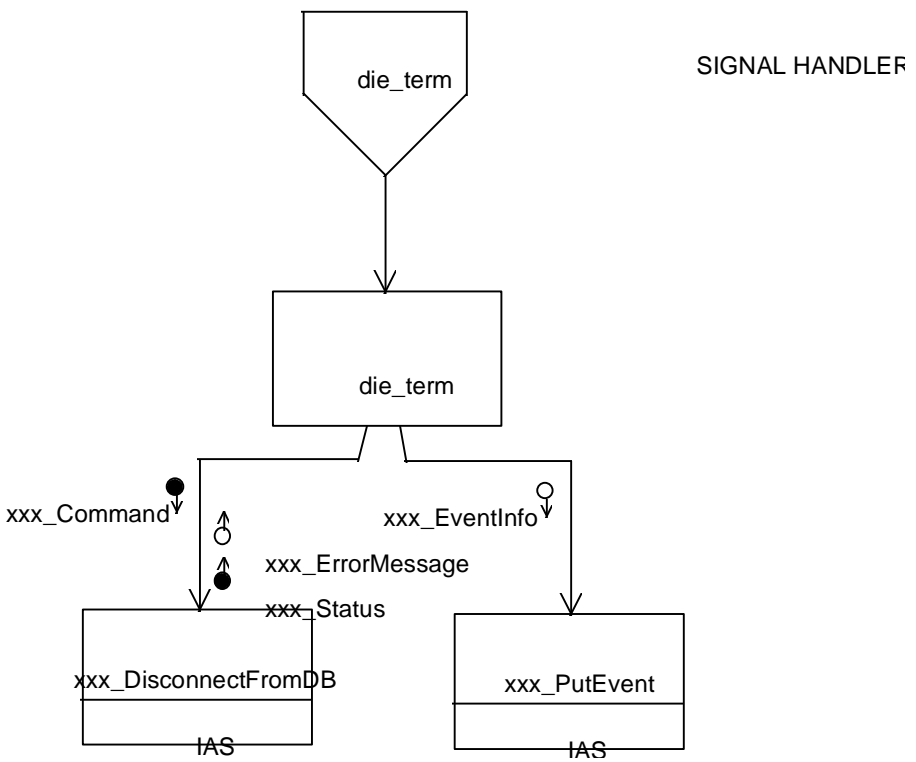


Figure 6-3. DIE Structure Chart (6 of 6)

## REVIEW

ENDDO

RETURN status and updated die\_pdr\_fnlist

### **die\_format\_data—Format Data From Product Request For Entry in Database**

#### **Parameters:**

die\_pdr\_filename : data\_in

die\_status : control\_out

**Body:** This module reads a product request file, extracts and formats the information then call ddb\_insert\_prod\_req to enter the data into the database.

### **die\_init—Initialize DIE Task**

#### **Parameters:**

die\_status : control\_out

die\_poll\_timer : data\_out

**Body:** The die\_init routine calls xxx\_ConnectToDB to connect to the database and retrieve configuration information. xdb\_get\_time\_period is called to get the polling time increment for die. The signal handler for termination is set up. If any errors have occurred, the error is logged using xxx\_LogError, and disconnects from the database using xxx\_DisconnectFromDB.

### **die\_main—Interface With ECS To Get Product Requests**

#### **Parameters:** None

**Body:** This module is responsible for retrieving, processing, and responding to Product Requests from ECS. It polls a specific directory on an ECS controlled server for the Product Request URFs. If a URF is accepted (response with PAN) the Product Request information is inserted in the database. This task also accepts, processes, and updates the database for cancel requests from the operator.

#### **PDL:**

CALL die\_init to initialize task, connect to the database and return timer increment

IF error return from die\_init THEN

EXIT

END IF

DO FOREVER

IF cancel request is received from operator THEN

CALL die\_process\_cancel

ENDIF

WAIT for timer event

CALL die\_poll\_for\_pdr

IF any product request files found THEN

CALL die\_eval\_urf to validate all files and insert valid Product Requests into

## REVIEW

```
        database and return filename list with validity status added
    CALL die_prepare_responses to build and send either PAN or PDRD for each
        PDR
    ENDIF
    SET timer for next event
ENDDO
EXIT
```

### **die\_poll\_for\_pdr—Poll ECS Server For Product Requests And Retrieve Them**

#### **Parameters:**

die\_status : control\_out  
die\_pdr\_fnlist : data\_out

**Body:** This module uses the EgFTP COTS software to access and perform polling and file retrieval from the ECS server. It uses the time stamp in the filenames to identify new files and FTP them to a local directory.

### **die\_prepare\_responses—Build And Send Response File (PAN or PDRD)**

#### **Parameters:**

die\_status : control\_out  
die\_pdr\_fnlist : data\_in

**Body:** This module creates and sends response files for received Product Requests. The response is either a PAN for a valid Product Request or a PDRD for a Product Request that contained errors. This module calls die\_create\_response to build the response file and then die\_send\_response to place the response file in the predefined directory for each entry in die\_pdr\_fnlist.

### **die\_process\_cancel—Respond to User Cancel Request**

#### **Parameters:**

L1\_Prod\_Cancel\_Req : control\_in  
die\_status : control\_out

**Body:** This module responds to a user request to cancel a product request. It calls xdb\_update\_prod\_req\_state to update the database.

### **die\_send\_response—Send Response File To ECS**

#### **Parameters:**

die\_resp\_filename : data\_in  
die\_status : control\_out

## REVIEW

**Body:** This module uses the EgFTP COTS software to FTP the response file (either a PAN or PDRD) to ECS. The response file is built by die\_create\_resp.

### **die\_term—DIE Termination Signal Handler**

**Parameters:** None

**Body:** This module terminates the DIE process. This module calls xxx\_DisconnectFromDB to disconnect from the database before termination.

### **6.3.2 DMS Ingest LOR Product (DIL) Task**

This subsection describes the DIL task software.

#### **6.3.2.1 Task Overview**

The DIL task is responsible for ingesting LOR image products from the ECS. The criteria used by this task for requesting LOR products from ECS may include disk space availability, prestaging of data for processing, and throttling to prevent overloading ECS. This task periodically requests the LOR products when the requirements for ingest have been met. This task then interfaces with the ECS to facilitate the transfer of the LOR product. This task also receives requests to stage the LOR products the operator has manually ingested.

#### **6.3.2.2 Initialization**

The task is started by the PSI task and accesses initialization data from the database for configuration.

#### **6.3.2.3 Normal Operation**

This task activates based on timer expiration or receipt of a user request. When the LOR request timer expires and the criteria for ingest have been met, this task sends ECS a request for additional LOR products on the ECS socket. The method for requesting the LOR products includes establishing a connection on the ECS socket, sending ECS a start session message, receiving a start session acknowledgment message from ECS, sending ECS the LOR data request message, receiving a LOR data request acknowledgment message from ECS, sending ECS a close session message, and closing the socket connection. After ECS has prepared the LOR product for transfer, the ECS notifies LPGS that the data is available by ftping a data availability notice (DAN) to a predefined directory on the LPGS. When the polling interval timer expires, the DIL task checks for new DAN messages and responds to the DAN by ftping a data availability acknowledgment (DAA) message into a predefined directory on ECS. Then, the DIL task creates the product directories, retrieves the LOR product from ECS via ftp, validates the LOR product, and catalogs the LOR product in the database. The DIL task then informs ECS that the product has been transferred by ftping a data delivery notice (DDN) message into a predefined directory on the ECS system. Upon receipt of the DDN, the ECS acknowledges the message by ftping a data delivery acknowledgment (DDA) message into a predefined directory on the LPGS. When the polling timer expires, the DIL task also checks for new DDA messages.

## REVIEW

When a new DDA message is received, the DIL task updates the database to indicate the LOR data is available for processing. When an operator request is received: the DIL task creates the product directories, moves the LOR product into the appropriate directory, validates the LOR product, catalogs the LOR product in the database, and updates the database to indicate the LOR data is available for processing.

### 6.3.2.4 Error Handling

The DIL task establishes a signal handler to capture all fatal UNIX signals which would cause the task to abort. The signal handler sends an error message to the operator, disconnects from the database, and gracefully shuts down the task. Nonfatal errors are logged to the LPGS event log and an error notification is sent to the user interface.

### 6.3.2.5 Design

This subsection presents the design of the DIL task. The structure chart for the task is illustrated in Figure 6-4. The module specifications for the DIL task are provided (in alphabetical order) below.

#### **dil\_build\_daa—DIL Build DAA**

##### **Parameters:**

dil\_dan\_stat : data\_in  
dil\_dan\_msg : data\_in  
dil\_msg\_data : data\_out

**Body:** This module builds the Data Availability Acknowledgment (DAA) message based on the inputted status.

#### **dil\_build\_data\_req—Build Data Request Message**

##### **Parameters:**

xxx\_prod\_req : data\_in  
dil\_msg\_data : data\_out  
xxx\_prod\_req\_id : data\_in

**Body:** This module builds the LOR data request message which is sent to ECS.

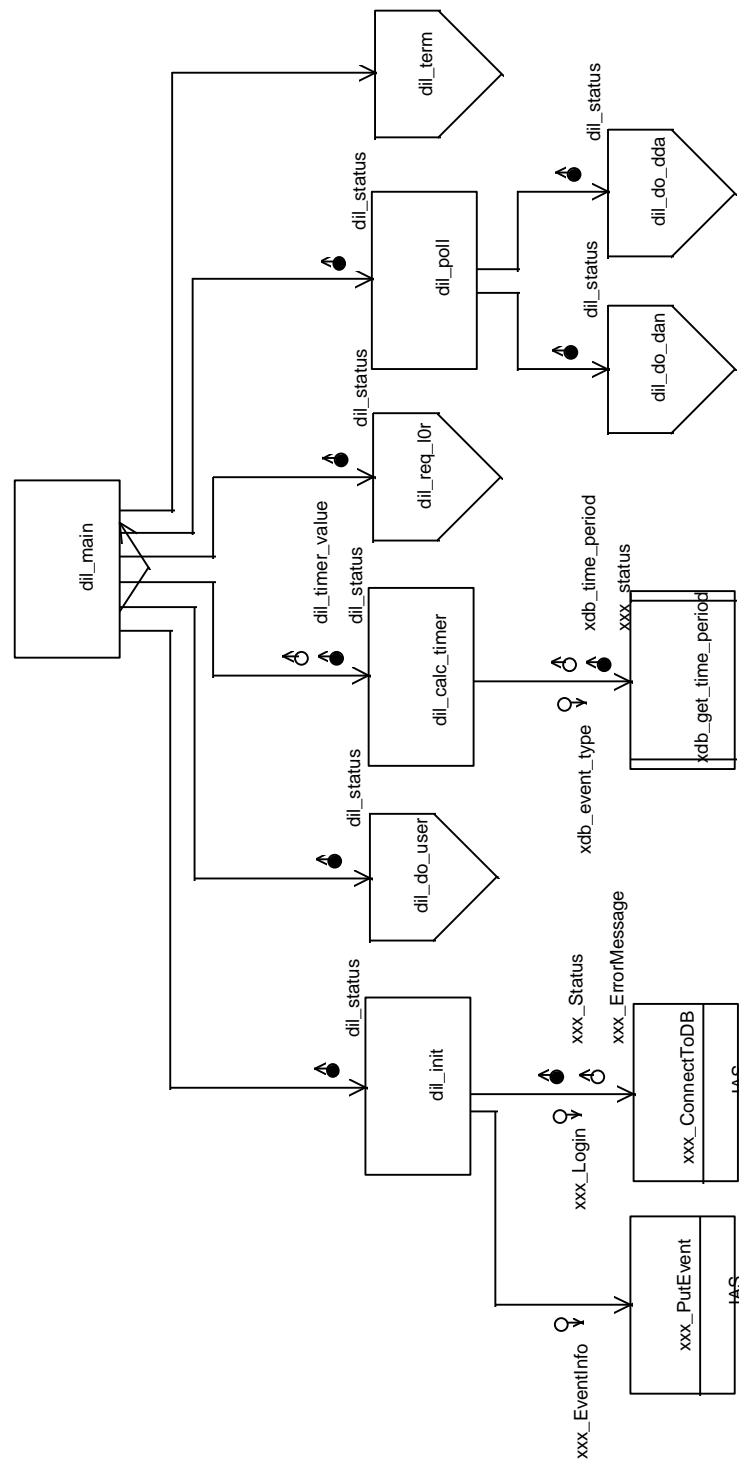
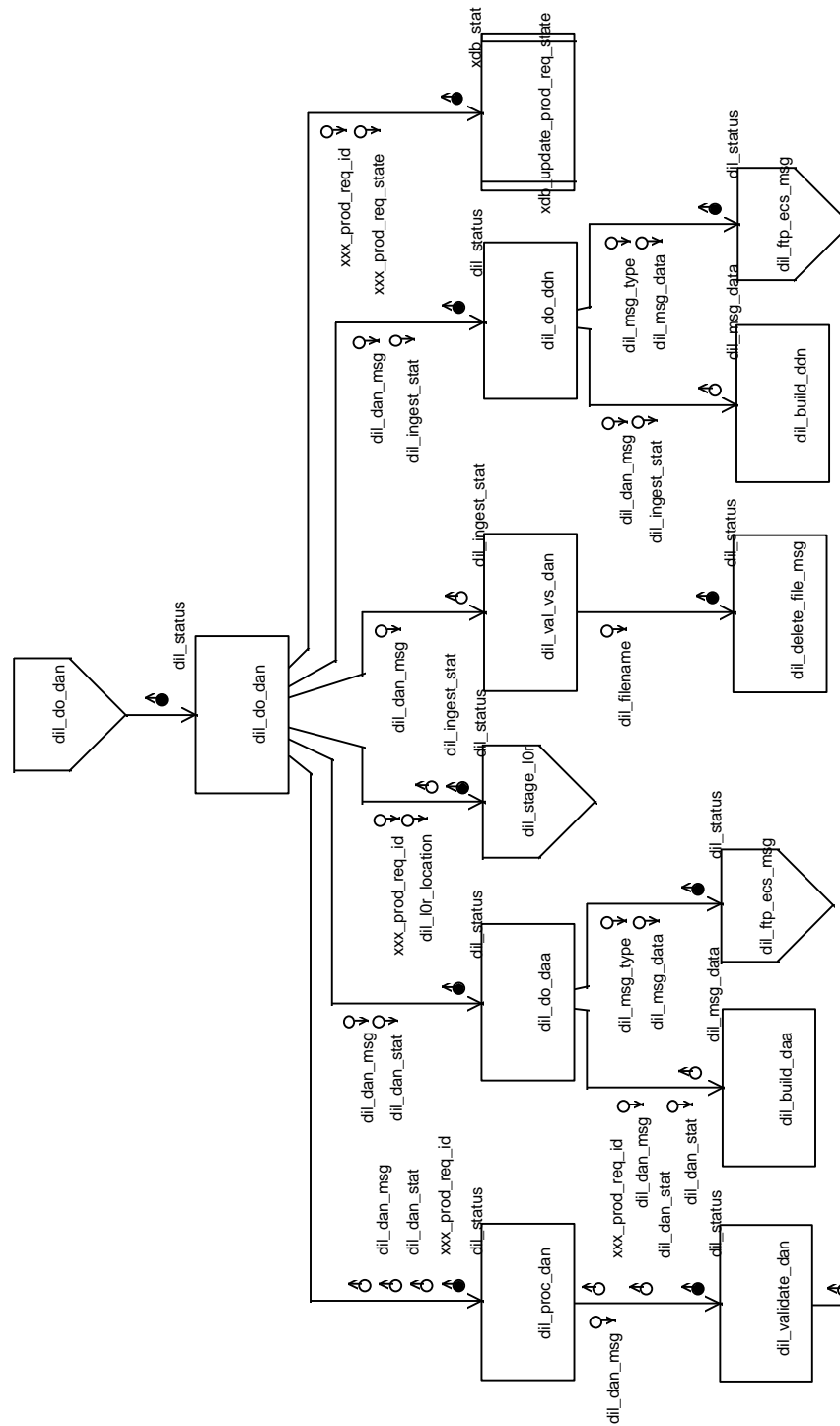


Figure 6-4. DIL Structure Chart (1 of 9)

## REVIEW



**Figure 6-4. DIL Structure Chart (2 of 9)**

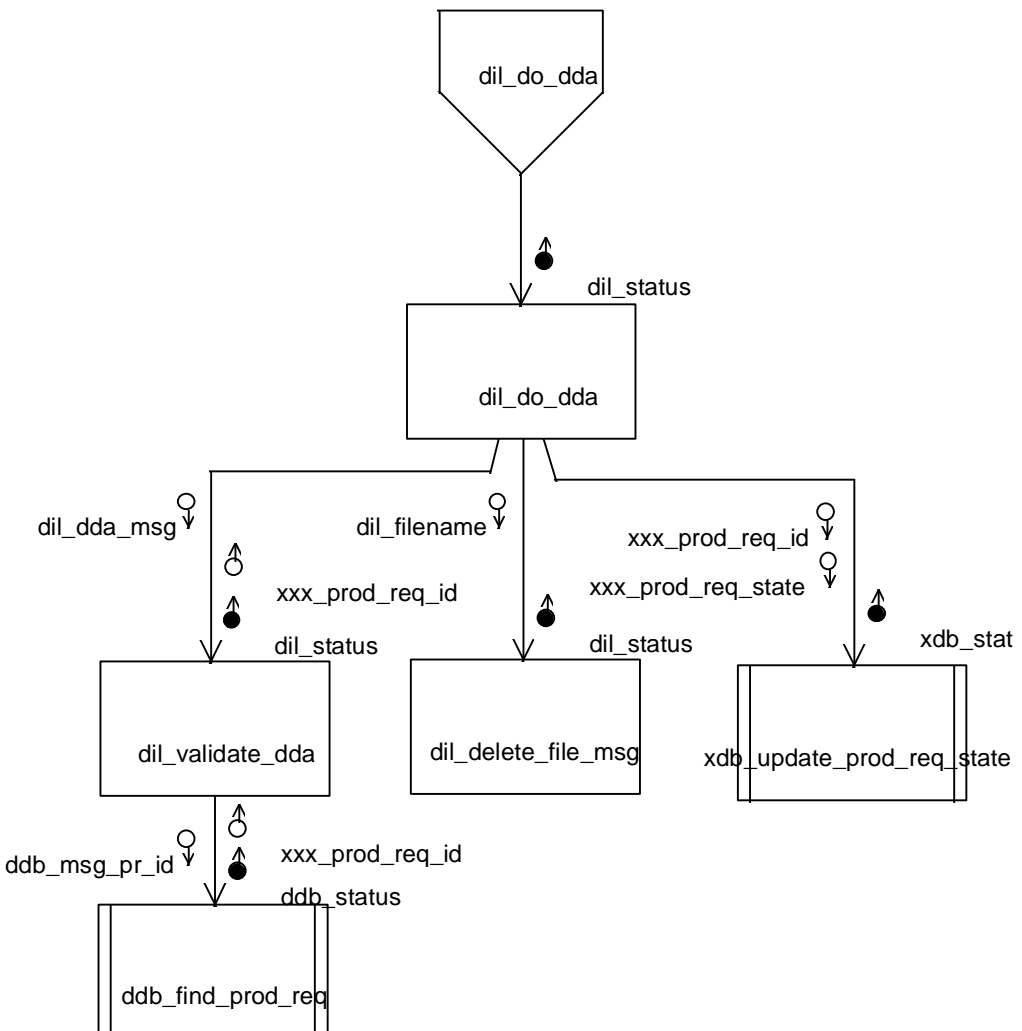


Figure 6-4. DIL Structure Chart (3 of 9)

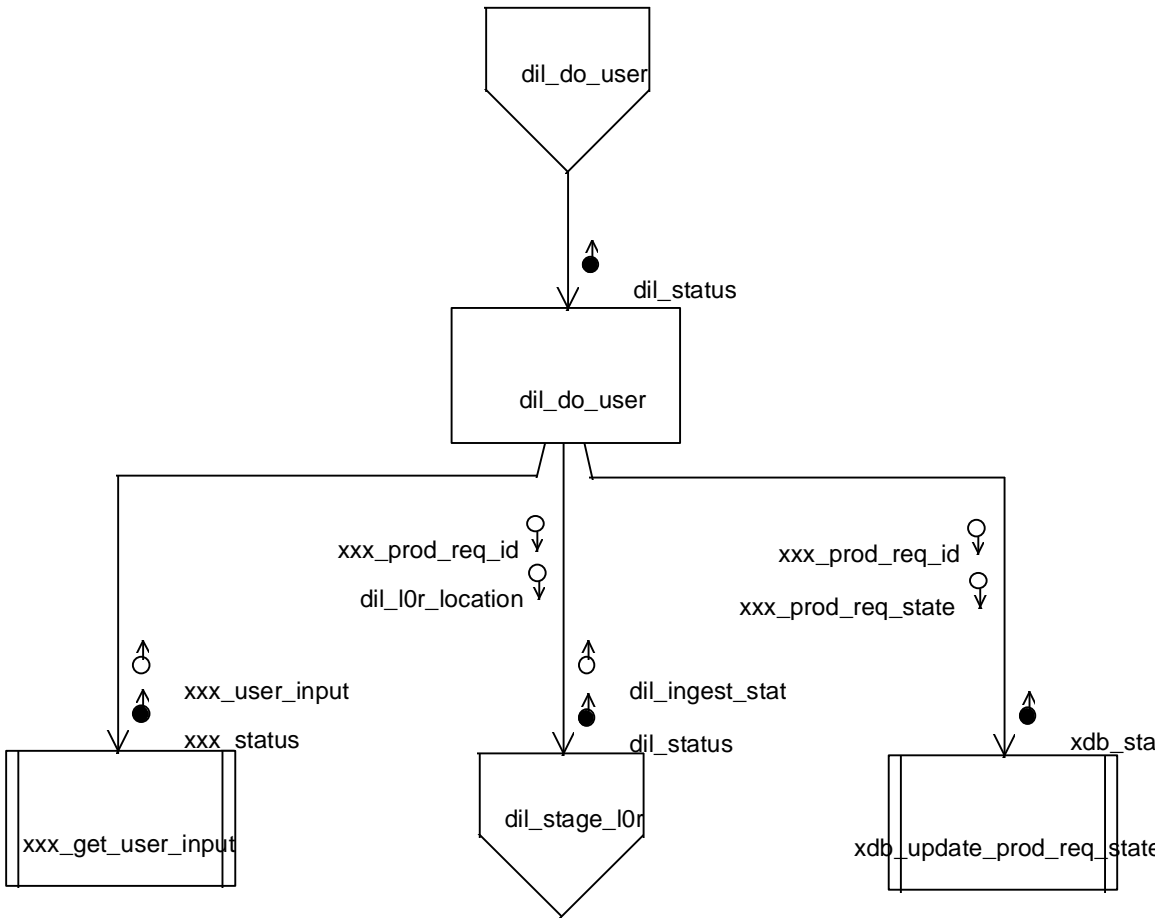


Figure 6-4. DIL Structure Chart (4 of 9)

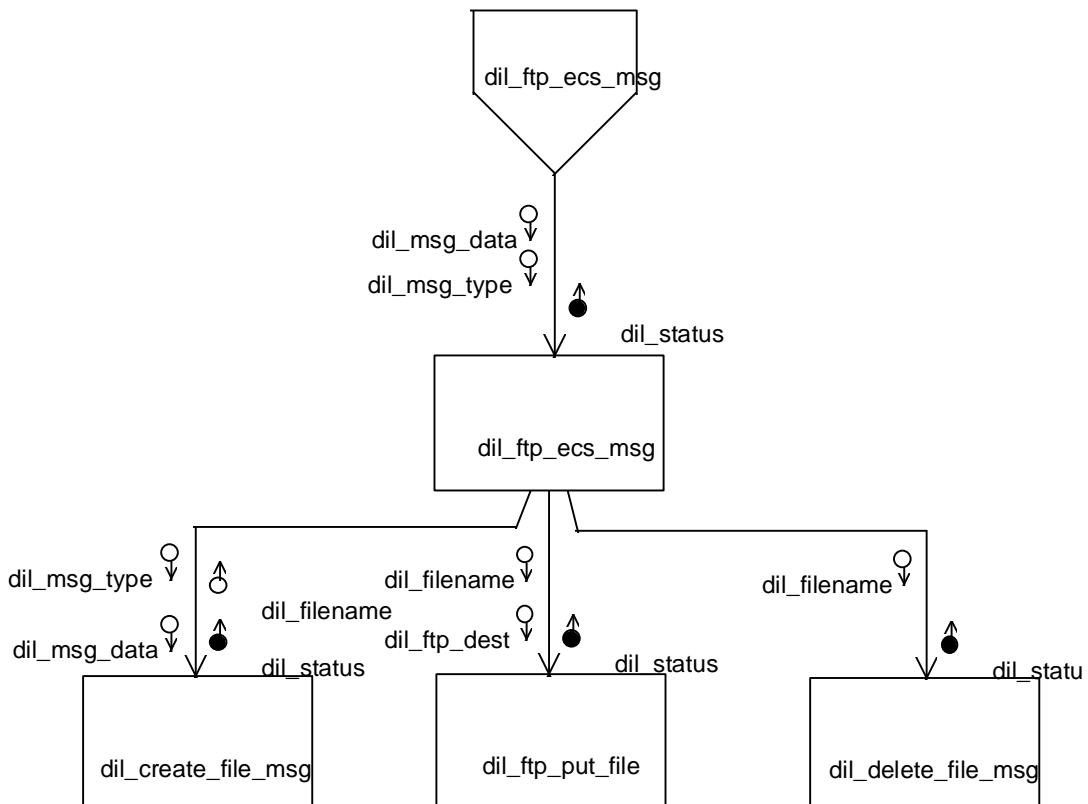


Figure 6-4. DIL Structure Chart (5 of 9)

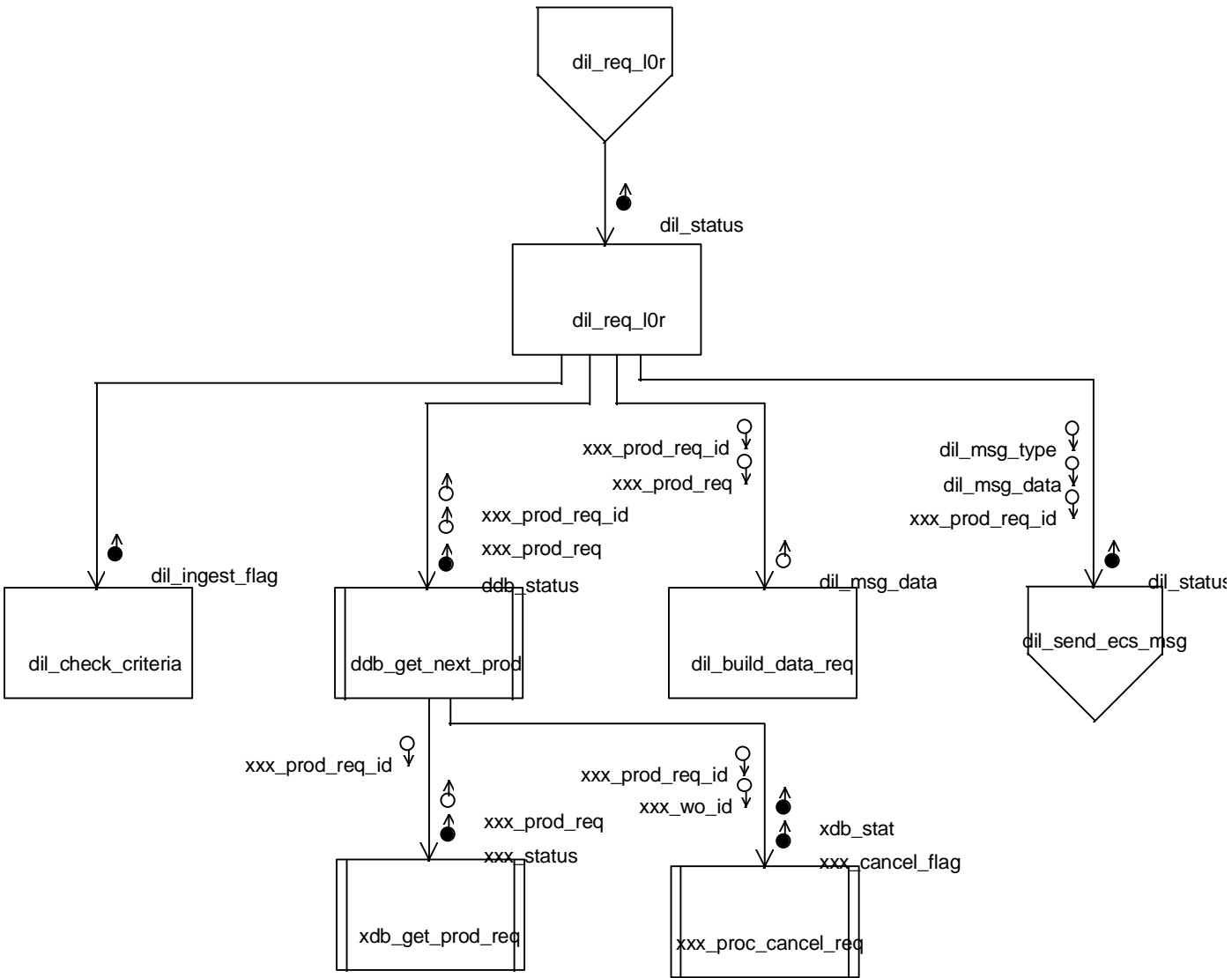
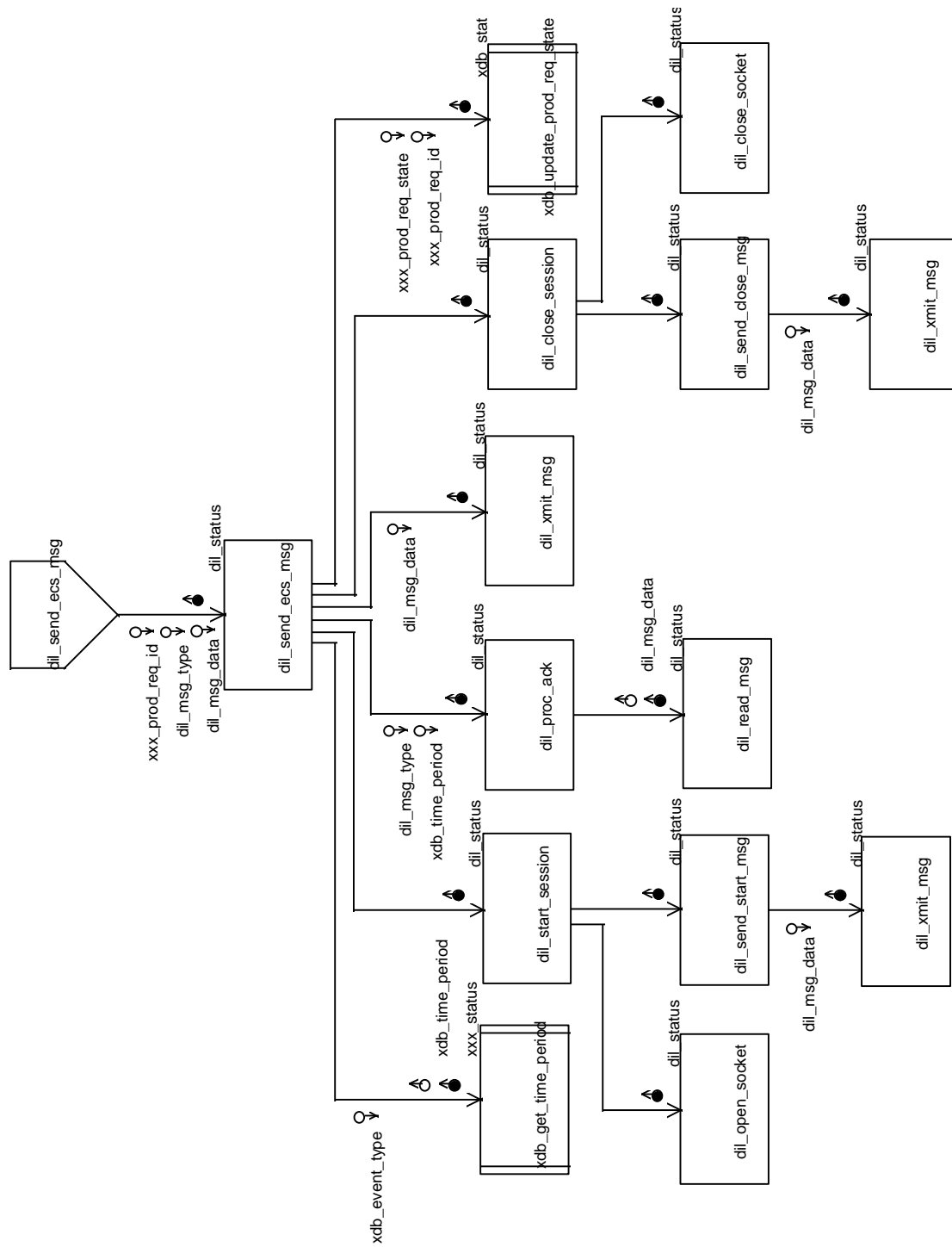


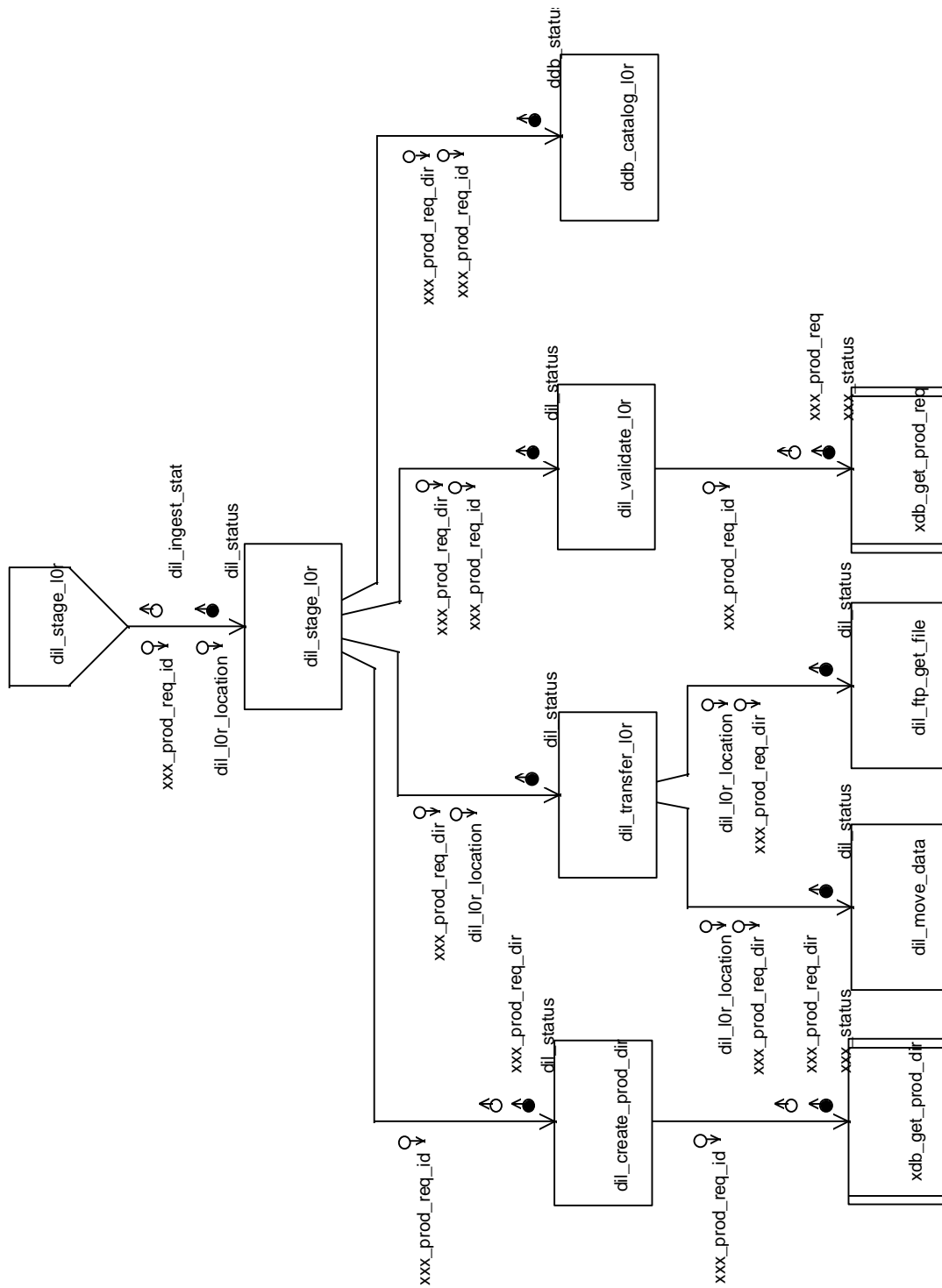
Figure 6-4. DIL Structure Chart (6 of 9)

## REVIEW



**Figure 6-4. DIL Structure Chart (7 of 9)**

## REVIEW



**Figure 6-4. DIL Structure Chart (8 of 9)**

The diagram illustrates the component structure and dependencies for the 'dil\_term' component. At the top, a component named 'dil\_term' is shown with a provided interface (pentagon shape). This component provides the 'dil\_term' interface (rectangle shape). Below this, two components are shown: 'xxx\_DisconnectFromDB' and 'xxx\_PutEvent'. The 'xxx\_DisconnectFromDB' component has a required interface (rectangle shape) and a provided interface (pentagon shape). The 'xxx\_PutEvent' component has a required interface (rectangle shape) and a provided interface (pentagon shape). The 'xxx\_DisconnectFromDB' component depends on the 'xxx\_DisconnectFromDB' interface provided by the 'dil\_term' component. The 'xxx\_PutEvent' component depends on the 'xxx\_PutEvent' interface provided by the 'dil\_term' component. The 'xxx\_DisconnectFromDB' component also provides the 'xxx\_DisconnectFromDB' interface. The 'xxx\_PutEvent' component also provides the 'xxx\_PutEvent' interface.

## REVIEW

### **dil\_build\_ddn—DIL Build DDN**

#### **Parameters:**

dil\_ingest\_stat : data\_in

dil\_dan\_msg : data\_in

dil\_msg\_data : data\_out

**Body:** This module builds the Data Delivery Notice message based on the inputted ingest status.

### **dil\_calc\_timer—Calculate Wakeup Time**

#### **Parameters:**

dil\_status : control\_out

dil\_timer\_value : data\_out

**Body:** The DIL process needs to periodically request L0R products from ECS. Also, the DIL process periodically polls the protocol directories to detect the receipt of message from ECS. The dil\_calc\_timer module determines which event will occur next. Then, this module calculates when the DIL process will need to wake up to process the next event.

### **dil\_check\_criteria—Check Ingest Criteria**

#### **Parameters:**

dil\_ingest\_flag : control\_out

**Body:** This module checks to see if the criteria for requesting Image Data from ECS have been met. The criteria may include: disk space availability; need to pre-stage Image data for processing; and need to prevent overloading ECS bandwidth. This module returns a flag indicating whether the criteria have been met or not.

### **dil\_close\_session—Close Session**

#### **Parameters:**

dil\_status : control\_out

**Body:** This module is called to terminate a session with the ECS system. First, this module calls dil\_send\_close\_msg to create and send the close session message to the ECS system via the socket. Second, this module calls dil\_close\_socket to close the socket with the ECS system.

### **dil\_close\_socket—Close A Socket**

#### **Parameters:**

dil\_status : control\_out

**Body:** This module calls the necessary system functions to close a socket on a remote system.

## REVIEW

### **dil\_create\_file\_msg—Create File Message**

#### **Parameters:**

dil\_status : control\_out  
dil\_filename : data\_out  
dil\_msg\_type : data\_in  
dil\_msg\_data : data\_in

**Body:** This module creates the protocol file message. This module creates and opens a temporary file. Then, this module writes the message data into the file. Then, this module closes the file. This module returns the name of the temporary file.

### **dil\_create\_prod\_dir—Create Product Directories**

#### **Parameters:**

xxx\_prod\_req\_id : data\_in  
dil\_status : control\_out  
xxx\_prod\_req\_dir : data\_out

**Body:** This module calls xdb\_get\_prod\_dir to get the name of the product request directory. Then, this module creates the product request directory. Next, this module creates the input and save subdirectories of the product request directory. This module returns the name of the product request directory.

**NOTE:** The following is the directory structure for the product requests:

- product\_request\_dir
  - input\_directory
    - L0R Product files in this directory
  - save\_directory
    - Files modified during processing are saved here
  - wo1\_directory
    - L1R & L1G Output files in this directory
  - wo2\_directory
    - L1R & L1G Output files in this directory

Each time a work order is run, a new WO directory is created.

The work order directories are created by PCS when the work is run.

### **dil\_delete\_file\_msg—Delete A File Message**

#### **Parameters:**

dil\_filename : data\_in  
dil\_status : control\_out

## REVIEW

**Body:** This module deletes the specified protocol file message.

### **dil\_do\_daa—DIL Do The DAA Message Processing**

**Parameters:**

dil\_dan\_stat : data\_in

dil\_dan\_msg : data\_in

dil\_status : control\_out

**Body:** After the receipt of a Data Availability Notice (DAN) message from ECS, LPGS responds to the message by sending a Data Availability Acknowledgment (DAA) message to ECS. This module does the DAA message processing. This module calls dil\_build\_daa to build the DAA message. Then, this module calls dil\_ftp\_ecs\_msg to ftp the DAA message to the ECS system.

### **dil\_do\_dan—Do The DAN Processing**

**Parameters:**

dil\_status : control\_out

**Body:** This module is called to do the DAN processing. After the receipt of a request for LOR data, ECS stages the data in the ingest directory. Then, ECS ftp puts a DAN message in the DAN protocol directory. This message signals the DIL process the LOR product is ready for ingest. This module is called when a new DAN is detected in the DAN protocol directory. (**NOTE:** It is assumed that ECS will send one DAN message at a time.) This module first calls dil\_proc\_dan to process the DAN message. Second, this module calls dil\_do\_daa to send ECS a response to the DAN message. Third, this module calls dil\_stage\_lor to stage the LOR data products in preparation for processing. Fourth, this module calls dil\_do\_ddn to inform ECS that the LOR data was delivered onto the LPGS system. After each of these steps in processing, this module calls xdb\_update\_prod\_req to update the state of the product request to indicate which processing is complete.

### **dil\_do\_dda—Do The DDA Processing**

**Parameters:**

dil\_status : control\_out

**Body:** This module is called to do the Data Delivery Acknowledgment (DDA) processing. After ECS receives a Data Delivery Notice from LPGS, ECS ftp puts a DDA message into the DDA protocol directory in response. This module is called when a new DDA is detected in the DDA protocol directory. This module first calls dil\_validate\_dda to validate the message. Then, this module calls dil\_delete\_file\_msg to delete the DDA file message. Then, this module calls xdb\_update\_prod\_req to update the state of the product request to indicate the LOR data is available for processing.

## REVIEW

### **dil\_do\_ddn—DIL Do The DDN Message Processing**

#### **Parameters:**

dil\_ingest\_stat : data\_in

dil\_dan\_msg : data\_in

dil\_status : control\_out

**Body:** After LPGS has successfully staged the LOR product, LPGS sends ECS a Data Delivery Notice (DDN) message to notify ECS that the data was received. This module does the DDN message processing. This module calls dil\_build\_ddn to build the DDN message based on the status of the ingest. Then, this module calls dil\_ftp\_ecs\_msg to ftp the DDN message to the ECS system.

### **dil\_do\_user—DIL Do The User Request**

#### **Parameters:**

dil\_status : control\_out

**Body:** The dil\_do\_user module is called to acknowledge the receipt of operator ingested LOR products. After the operator manually ingests the LOR products, a command is sent to the DIL process via the user interface. This module calls xxx\_get\_user\_input to get the command. Then, this module calls dil\_stage\_lor to stage the LOR products in preparation for processing. Finally, this module calls xdb\_update\_prod\_req\_state to update the product request state to indicate the data is available for processing.

### **dil\_ftp\_ecs\_msg—DIL FTP ECS Messages**

#### **Parameters:**

dil\_msg\_data : data\_in

dil\_msg\_type : data\_in

dil\_status : control\_out

**Body:** This module ftp puts protocol messages onto the ECS system. This module calls dil\_create\_file\_msg to create the protocol file message. Then, this module calls dil\_ftp\_put\_file to ftp put the file onto the ECS system. Finally, this module calls dil\_delete\_file\_msg to delete the protocol file message from the LPGS system.

### **dil\_ftp\_get\_file—DIL FTP Get File**

#### **Parameters:**

xxx\_prod\_req\_dir : data\_in

dil\_lor\_location : data\_in

dil\_status : control\_out

## REVIEW

**Body:** This module uses Eg\_FTP COTS library routines to ftp get the LOR products from the ECS ingest directory. The LOR product is ftped into the appropriate product request input directory.

### **dil\_ftp\_put\_file—DIL FTP Put A File**

**Parameters:**

dil\_ftp\_dest : data\_in  
dil\_status : control\_out  
dil\_filename : data\_in

**Body:** This module uses Eg\_FTP COTS library routines to ftp put a file to the specified destination.

### **dil\_init—DIL Initialization**

**Parameters:**

dil\_status : control\_out

**Body:** The dil\_init module calls xxx\_ConnectToDB to connect to the database. Then, this module sets up a signal handler for process termination. If an error occurs during processing, this module calls xxx\_PutEvent to report the error and returns a bad status.

### **dil\_main—DMS Ingest LOR Product Main Module**

**Parameters:** None

**Body:** dil\_main is the main module for the DMS Ingest LOR Product process. This module calls the dil\_init to initialize the process. Then, this module calls dil\_calc\_timer to determine the next time the process will need to wake up. Then, dil\_main waits for the timer to expire or for the receipt of a message from the operator. When the timer expires, this module either calls dil\_req\_l0r to request additional LOR products or dil\_poll which polls the protocol directories for messages. When an operator request is received, dil\_main calls dil\_do\_user to acknowledge operator ingested data.

**PDL:**

```
CALL dil_init to initialize the process
DOWHILE dil_status indicates processing should continue
  CALL dil_calc_timer to determine when to wake up
  IF dil_status is good THEN
    SET the timer to expire in dil_timer_value time
    WAIT for the timer to expire or for input from the user
    IF the timer expired THEN
      IF it is time to request additional LOR products THEN
        CALL dil_req_l0r to request the products from ECS
      ELSE
```

## REVIEW

```
        IF it is time to poll the protocol directories THEN
            CALL dil_poll to poll the directories for messages
        ENDIF
    ENDIF
ENDIF
IF process was notified of user input THEN
    CALL dil_do_user to process the user input
ENDIF
ENDIF
ENDDO
CALL dil_term to terminate the dil_main process
EXIT
```

### **dil\_move\_data—DIL Move Data**

#### **Parameters:**

xxx\_prod\_req\_dir : data\_in  
dil\_l0r\_location : data\_in  
dil\_status : control\_out

**Body:** This module is called when the operator has manually ingested LOR products. This module moves the operator ingested LOR products to the appropriate product request input directory.

### **dil\_open\_socket—Open A Socket**

#### **Parameters:**

dil\_status : control\_out

**Body:** This module calls the necessary system functions to open a socket on a remote system.

### **dil\_poll—DIL Poll Protocol Directories**

#### **Parameters:**

dil\_status : control\_out

**Body:** The dil\_poll module is called periodically to poll the protocol directories. This module checks the DAN and DDA directories for the receipt of new messages from ECS. If a new message is detected in the DAN directory, this module calls dil\_do\_dan to process the message. And, if a new message is detected in the DDA directory, this module calls dil\_do\_dda to process the message.

## REVIEW

### **dil\_proc\_ack—Process Acknowledgment Message**

#### **Parameters:**

dil\_msg\_type : data\_in  
dil\_status : control\_out  
xdb\_time\_period : data\_in

**Body:** After sending ECS a start session message or a LOR data request message on the socket, ECS should respond with an acknowledgment message. This module first set a timer for the specified time-out period. Then, this process waits for the timer to expire or for the receipt of the acknowledgment message. If the acknowledgment message is received, this module calls dil\_read\_msg to get and validate the message. If the acknowledgment message was invalid or was not received in time, this module returns an error.

### **dil\_proc\_dan—Process DAN Message**

#### **Parameters:**

dil\_dan\_stat : data\_out  
dil\_status : control\_out  
dil\_dan\_msg : data\_out  
xxx\_prod\_req\_id : data\_out

**Body:** This module processes the DAN message. This module opens and reads the DAN file message in the DAN directory. Then, this module calls dil\_validate\_dan to validate the DAN message.

### **dil\_read\_msg—Read A Message**

#### **Parameters:**

dil\_status : control\_out  
dil\_msg\_data : data\_out

**Body:** This module reads a message from the ECS socket.

### **dil\_req\_l0r—Request LOR Products**

#### **Parameters:**

dil\_status : control\_out

**Body:** This module is called to request LOR products from ECS. This module calls dil\_check\_criteria to determine if LPGS can ingest additional LOR product. If LPGS is ready for additional LOR products, this module calls a series of modules to request the LOR products. First, dil\_req\_l0r calls ddb\_get\_next\_prod to determine which product request needs LOR data.

## REVIEW

Second, `dil_req_l0R` calls `dil_build_data_req` to build the L0R data request message. Third, this module calls `dil_send_ecs_msg` to send the L0R data request message to ECS.

### **`dil_send_close_msg`—Send Close Session Message**

#### **Parameters:**

`dil_status` : control\_out

**Body:** This module first creates the close session message which has to be sent to ECS. Then, this module calls `dil_xmit_msg` to transmit the close session message to ECS via the socket.

### **`dil_send_ecs_msg`—Send A Message To ECS**

#### **Parameters:**

`dil_msg_data` : data\_in

`dil_msg_type` : data\_in

`xxx_prod_req_id` : data\_in

`dil_status` : control\_out

**Body:** This module sends messages to ECS. First, this module calls `xdb_get_time_period` to determine how long to wait for ECS to acknowledge the receipt of a message. Then, this module calls `dil_start_session` to start a session with the ECS system. This module then calls `dil_proc_ack` to wait for ECS to send the start session acknowledgment message. Next, this module calls `dil_xmit_msg` to transmit the inputted message to ECS. Next, this module calls `dil_proc_ack` to wait for and process the acknowledgment message ECS should send in response to the message. Finally, this module calls `dil_close_session` to terminate the session with the ECS system. This module also calls `xdb_update_prod_req_state` to record the receipt and transmission of each of the protocol messages.

#### **PDL:**

```
CALL xdb_get_time_period to determine how long to wait for ack messages from ECS
CALL dil_start_session to start the session with ECS
IF dil_start_session didn't return an error THEN
    CALL xdb_update_prod_req_state to indicate the start session message was sent
    CALL dil_proc_ack to wait for the ECS ack message
    IF dil_proc_ack didn't return an error THEN
        CALL xdb_update_prod_req_state to indicate the start session ack message was
        received
        CALL dil_xmit_msg to transmit the inputted message
        IF dil_xmit_msg didn't return an error THEN
            CALL xdb_update_prod_req_state to indicate the message was sent
            CALL dil_proc_ack to wait for the ECS ack message
            IF dil_proc_ack didn't return an error THEN
                CALL xdb_update_prod_req_state to indicate the ack message was
```

## REVIEW

```
        received
    CALL dil_close_session to close the session
    IF dil_close_session didn't return an error THEN
        CALL xdb_update_prod_req_state to indicate the session was
        terminated
    ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
NOTE: IF an error occurred, take the appropriate action as defined in the ECS/LPGS
Interface concepts working paper.
```

### **dil\_send\_start\_msg—Send A Start Message To ECS**

#### **Parameters:**

dil\_status : control\_out

**Body:** This module first creates the start session message which has to be sent to ECS. Then, this module calls dil\_xmit\_msg to transmit the start session message to ECS on a socket.

### **dil\_stage\_l0r—DIL Stage L0R Products**

#### **Parameters:**

dil\_l0r\_location : data\_in

xxx\_prod\_req\_id : data\_in

dil\_status : control\_out

dil\_ingest\_stat : data\_out

**Body:** This module is called to stage the L0R products in preparation for processing. First, dil\_stage\_l0r calls dil\_create\_prod\_dir to create the product request input directories. Then, this module calls dil\_transfer\_l0r to transfer the L0R products to the product request input and save directories. Next, dil\_stage\_l0r calls dil\_validate\_l0r to validate the L0R products and ddb\_catalog\_l0r to catalog the L0R products.

### **dil\_start\_session—Start A Session With ECS**

#### **Parameters:**

dil\_status : control\_out

**Body:** This module is called to start a session with the ECS system. First, dil\_start\_session calls dil\_open\_socket to open the socket with the ECS system. Second, this module calls dil\_send\_start\_msg to create and send the start session message to the ECS system on the opened socket.

## REVIEW

### **dil\_term—Terminate The DIL Process**

**Parameters:** None

**Body:** This module terminates the DIL process. This module calls xxx\_DisconnectFromDB to disconnect from the database. Next, this module will close the socket to ECS if it was open. Finally, this module terminates the DIL process.

### **dil\_transfer\_l0r—DIL Transfer LOR Product**

**Parameters:**

dil\_l0r\_location : data\_in

dil\_status : control\_out

xxx\_prod\_req\_dir : data\_in

**Body:** This module transfers the LOR products to the product request directories. First, this module determines if the product is on the LPGS system (manually ingested) or is on the ECS system. If the LOR product is on the LPGS system, this module calls dil\_move\_data to move the data to the product request input directory. If the LOR Product is on ECS, this module calls dil\_ftp\_get\_file to ftp the data onto the LPGS system into the product request input directory. Finally, dil\_transfer\_l0r moves the LOR products which are destroyed during processing, to the product request save directory.

### **dil\_val\_vs\_dan—Validate Data Received Versus DAN Message**

**Parameters:**

dil\_ingest\_stat : data\_out

dil\_dan\_msg : data\_in

**Body:** After the LOR product files are ftped onto the LPGS system, this module is called to make sure the correct files were transferred. This module validates the data file received against what file are specified in the Data Availability Notice (DAN). This module compares the list of files specified in the DAN message against the files located in the input directory. This module returns an error if any of the files specified in the DAN message are missing.

### **dil\_validate\_dan—Validate DAN**

**Parameters:**

dil\_dan\_stat : data\_out

dil\_dan\_msg : data\_in

dil\_status : control\_out

xxx\_prod\_req\_id : data\_out

## REVIEW

**Body:** This module validates the DAN message. This module extracts the product request id from the DAN message. Then, this module calls `ddb_find_prod_req` to try and find the corresponding product request. Next, this module validate the DAN message. If the product request isn't found or if the DAN message is invalid, this module returns an error.

### **dil\_validate\_dda—Validate DDA Message**

#### **Parameters:**

`xxx_prod_req_id` : data\_out

`dil_dda_msg` : data\_in

`dil_status` : control\_out

**Body:** This module validates the Data Delivery Acknowledgment (DDA) message. This module extracts the product request id from the DDA message. Then, this module calls `ddb_find_prod_req` to try and find the corresponding product request. Next, this module validates the DDA message. If the product request isn't found or if the DDA message is invalid, this module returns an error.

### **dil\_validate\_l0r—Validate The L0R Product Files**

#### **Parameters:**

`xxx_prod_req_id` : data\_in

`xxx_prod_req_dir` : data\_in

`dil_status` : control\_out

**Body:** This module checks the product request directory files. This module verifies all of the files needed for processing are present (L0R image, Cal Parm File, PCD, MSCD, ...). Next, this module calls `xdb_get_prod_req` to get the product request from the database. Then, this module checks to ensure the correct L0R image file for the product request is in the input product request directory. (re: product request was for Baltimore, received image file for Washington)

### **dil\_xmit\_msg—Transmit A Message To ECS**

#### **Parameters:**

`dil_msg_data` : data\_in

`dil_status` : control\_out

**Body:** This module transmits a message to ECS via the open socket.

### **6.3.3 DMS Process L0R Product (DPL) Task**

This subsection describes the DPL task software.

## REVIEW

### 6.3.3.1 Task Overview

The DPL task, which runs under the control of a script initiated by PCS, preprocesses the LOR image files. This task validates the LOR product and appends the results in a trending data table in the LPGS database. DPL also generates the consensus PCD and MSCD files.

### 6.3.3.2 Initialization

The DPL task is started by the PWC task and is passed an ODL filename as a parameter. It then retrieves all initialization data after processing the ODL file.

### 6.3.3.3 Normal Operation

This task runs when started by PCS. It processes and validates the image and non-image data, updates the database, generates the consensus PCD and MSCD, and returns processing status.

### 6.3.3.4 Error Handling

Nonfatal errors are logged to the LPGS event log and an error notification is sent to the database.

### 6.3.3.5 Design

This subsection presents the design of the DPL task. The structure chart for the task is illustrated in Figure 6-5. The module specifications for the DPL task are provided (in alphabetical order) below.

#### **dpl\_copy\_save\_to\_input—Copy LOR from Save to Input Directory Driver Module**

##### **Parameters:**

dpl\_status : control\_out

dpl\_prod\_req\_dir : data\_in

**Body:** This module copies the “saved” LOR product (those files are being destroyed during L1 processing) from the product request “save” sub-directory to the product request “input” sub-directory.

**NOTE:** Before L1 processing starts, copies of a selected subgroup of LOR product that will be modified during the L1 processing are kept in a “save” directory. Should the LOR product be reprocessed, these files can be restored from the “save” area to keep LOR integrity.

#### **dpl\_init—DPL Initialization Driver Module**

##### **Parameters:**

dpl\_prod\_req\_dir : data\_out

dpl\_status : control\_out

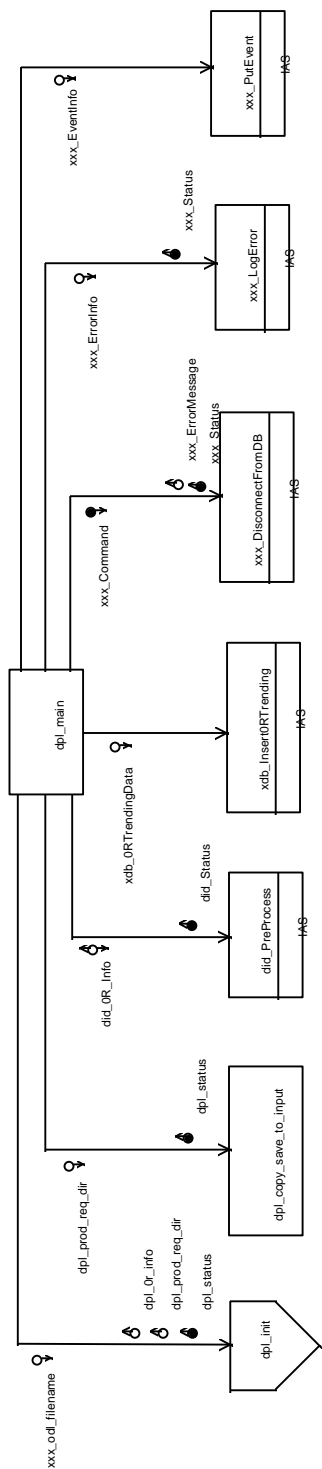


Figure 6-5. DPL Structure Chart (1 of 2)

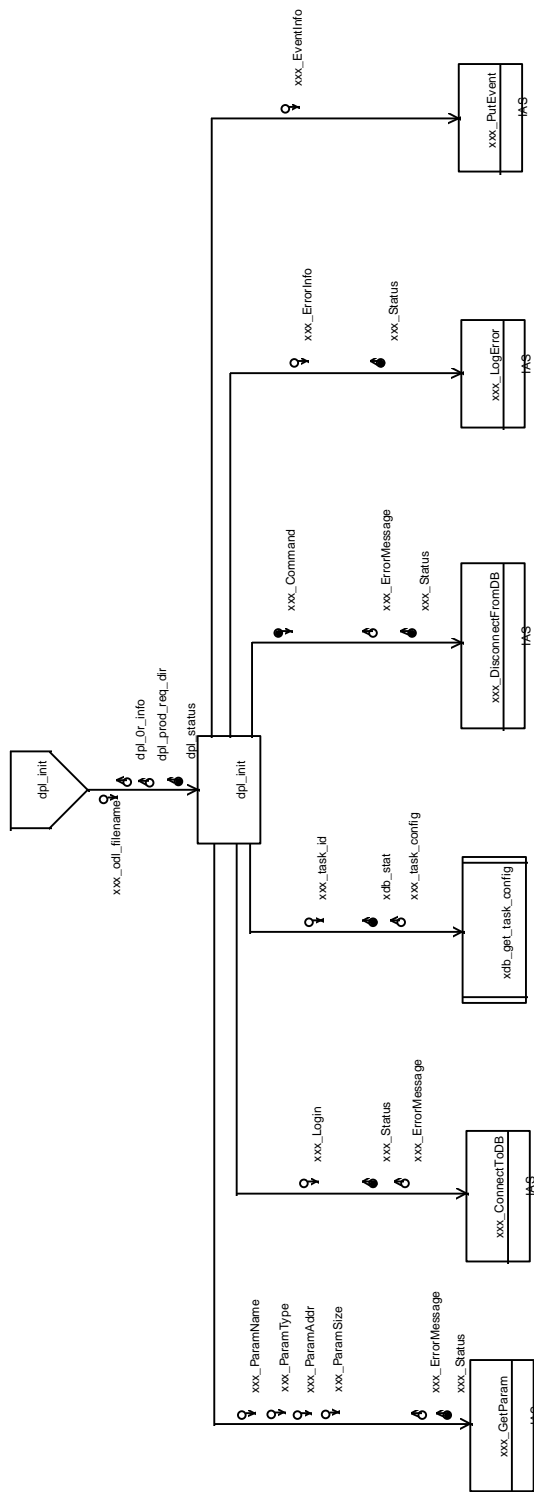


Figure 6-5. DPL Structure Chart (2 of 2)

## REVIEW

xxx\_odl\_filename : data\_in

dpl\_0r\_info : data\_out

**Body:** dpl\_init initializes the DPL task. It calls xxx\_GetParam to process the input script parameters, xxx\_odl\_filename, which points to an ODL file containing product request file directories and other attribute information. It then connects to the LPGS database. dpl\_init can also retrieve from the database all configuration information that are not provided by the ODL file.

### PDL:

```
CALL xxx_ConnectToDB to open access to the LPGS database
CALL xxx_GetParam to access the ODL file given by xxx_odl_filename to obtain
    dpl_prod_req_dir, file locations associated with specific product request, and other
    configuration information
IF error occurs in xxx_GetParam THEN
    CALL xxx_LogError to report to error log
    CALL xxx_PutEvent to post event to database
    CALL xxx_DisconnectFromDB to disconnect the LPGS database.
    RETURN error dpl_status
ENDIF
IF more configuration information are needed THEN
    CALL xdb_get_task_config to retrieve needed configuration parameters from the
    LPGS database
ENDIF
IF error in database access THEN
    CALL xxx_LogError to report to error log
    CALL xxx_PutEvent to post event to database
    CALL xxx_DisconnectFromDB to disconnect the LPGS database
    RETURN error dpl_status
ENDIF
RETURN
```

### dpl\_main—DPL Driver Module

**Parameters:** None

**Body:** dpl\_main first calls dpl\_init to initialize the DPL task based on information passed to the task and information in the database. It then calls dpl\_copy\_save\_to\_input to copy a select-to-be-saved subgroup of LOR product from “save” directory to “input” directory. Next it calls did\_PreProcess to check the LOR product to ensure that processing criteria are met. It then calls xdb\_InsertORTrending to store the validation and quality check results in a trending data table in the LPGS database. Finally it calls xxx\_DisconnectFromDB to disconnect the database.

**NOTE:** The IAS module, did\_PreProcess, is assumed to possess a decent algorithm for the generation of consensus PCD & MSCD. LPGS developer should pay attention to it while reusing this module.

## REVIEW

### PDL:

```
CALL dpl_init to process information passed via ODL file and retrieve information in
the database to initialize the DPL task
IF error in dpl_init THEN
    SET error return status for dpl_main
    EXIT
ENDIF
CALL dpl_copy_save_to_input to copy the part of “saved” L0R product from the
“save” directory to the “input” directory, using configuration information in
dpl_or_info
IF error in dpl_copy_save_to_input THEN
    SET error return status for dpl_main
    CALL xxx_LogError to write to error log
    CALL xxx_PutEvent to post event in database
ELSE
    CALL did_PreProcess to validate scene image data, CPF data, geo location table,
    SLO file, HDF directory, and IC data; to validate and generate consensus PCD
    plus MSCD data; and to create DDR and L0R preprocess report
    IF error in did_PreProcess THEN
        SET error return status for dpl_main
        CALL xxx_LogError to write to error log
        CALL xxx_PutEvent to post event in database
    ELSE
        CALL xdb_Insert0RTrending to write the preprocess results to the “trending
        data table” in the LPGS database
        IF error in accessing database or in generating the trending data THEN
            SET error return status for dpl_main
            CALL xxx_LogError to write to error log
            CALL xxx_PutEvent to post event in database
        ENDIF
    ENDIF
ENDIF
CALL xxx_DisconnectFromDB to disconnect the LPGS database
EXIT
```

### 6.3.4 DMS Format L1 Product (DFL) Task

This subsection describes the DFL task software.

#### 6.3.4.1 Task Overview

The DFL task is responsible for formatting and packaging the L1 product in preparation for delivery to ECS.

## REVIEW

### 6.3.4.2 Initialization

The DFL task is started by the PWC task and is passed an ODL filename as an input parameter. It then retrieves all initialization data after processing the ODL file.

### 6.3.4.3 Normal Operation

DFL is invoked by the PCS Work Order Controller with an ODL filename. After initialization, DFL formats the L1 product based on parameters in the ODL file. The L1 product contains L1R or L1G image data, metadata, and ancillary data, which get formatted and packaged by default into HDF-EOS format. Upon user specification, DFL could also format the L1G product to either GeoTIFF or FAST-C format. It then moves the L1 product to a designated delivery directory in preparation for delivery to ECS and returns processing status to PCS.

### 6.3.4.4 Error Handling

Nonfatal errors are logged to the LPGS event log and an error notification is sent to the database.

### 6.3.4.5 Design

This subsection presents the design of the DFL task. The structure chart for the task is illustrated in Figure 6-6. The module specifications for the DFL task are provided (in alphabetical order) below.

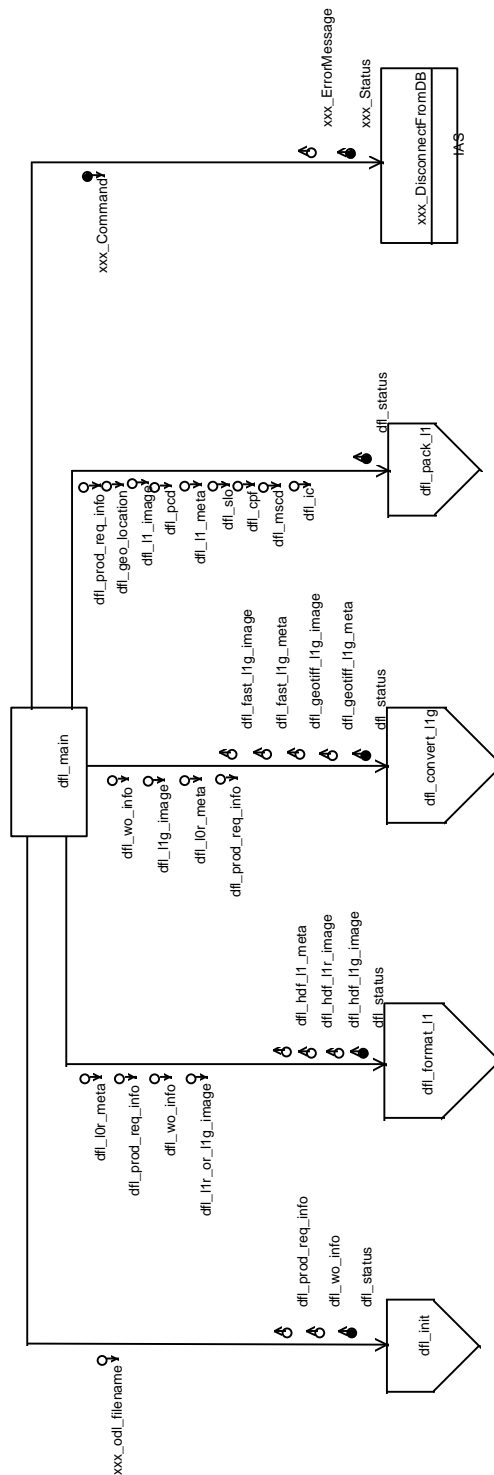
#### **df1\_assemble\_l1g\_prod—Assemble L1G Product**

##### **Parameters:**

df1\_status : control\_out  
df1\_l1g\_product : data\_out  
df1\_geotiff\_l1g\_image : data\_in  
df1\_geotiff\_l1g\_meta : data\_in  
df1\_fast\_l1g\_meta : data\_in  
df1\_hdf\_l1g\_image : data\_in  
df1\_hdf\_l1g\_meta : data\_in  
df1\_fast\_l1g\_image : data\_in

**Body:** df1\_assemble\_l1g\_prod packs the final L1G product together. It retrieves the components of the L1G product based on the product request specification, labels each file according to naming convention and moves (or copies) them into one area (directory). Generally, L1G product should includes L1G image, and meta data file. They can be in any of the following three formats: HDF-EOS, FAST-C, or GeoTIFF.

## REVIEW



**Figure 6-6. DFL Structure Chart (1 of 5)**

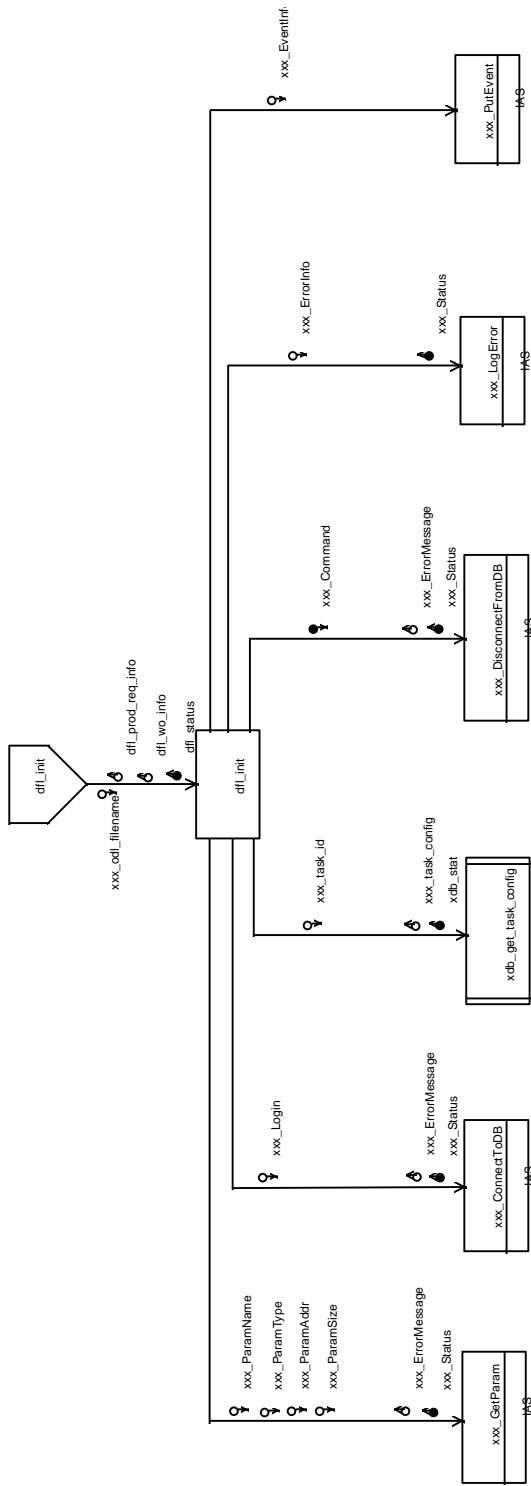
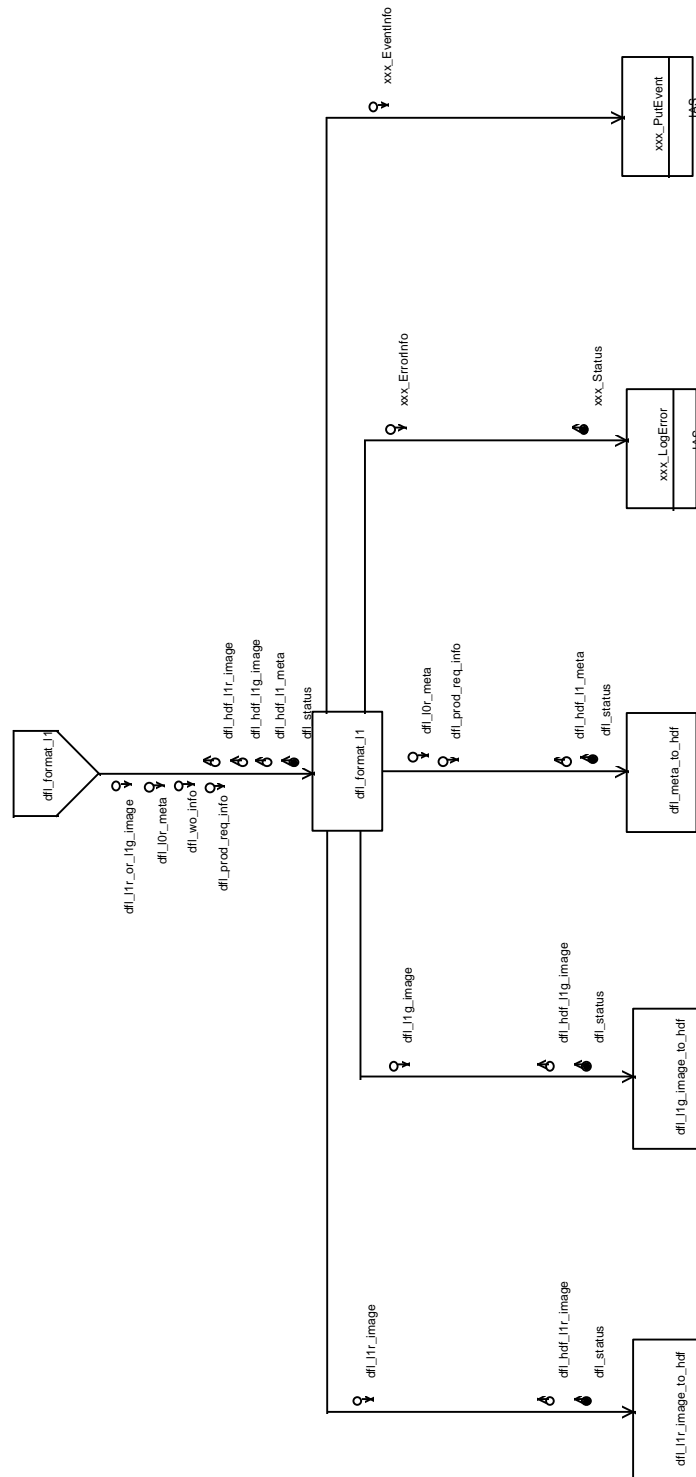


Figure 6-6. DFL Structure Chart (2 of 5)

## REVIEW



**Figure 6-6. DFL Structure Chart (3 of 5)**

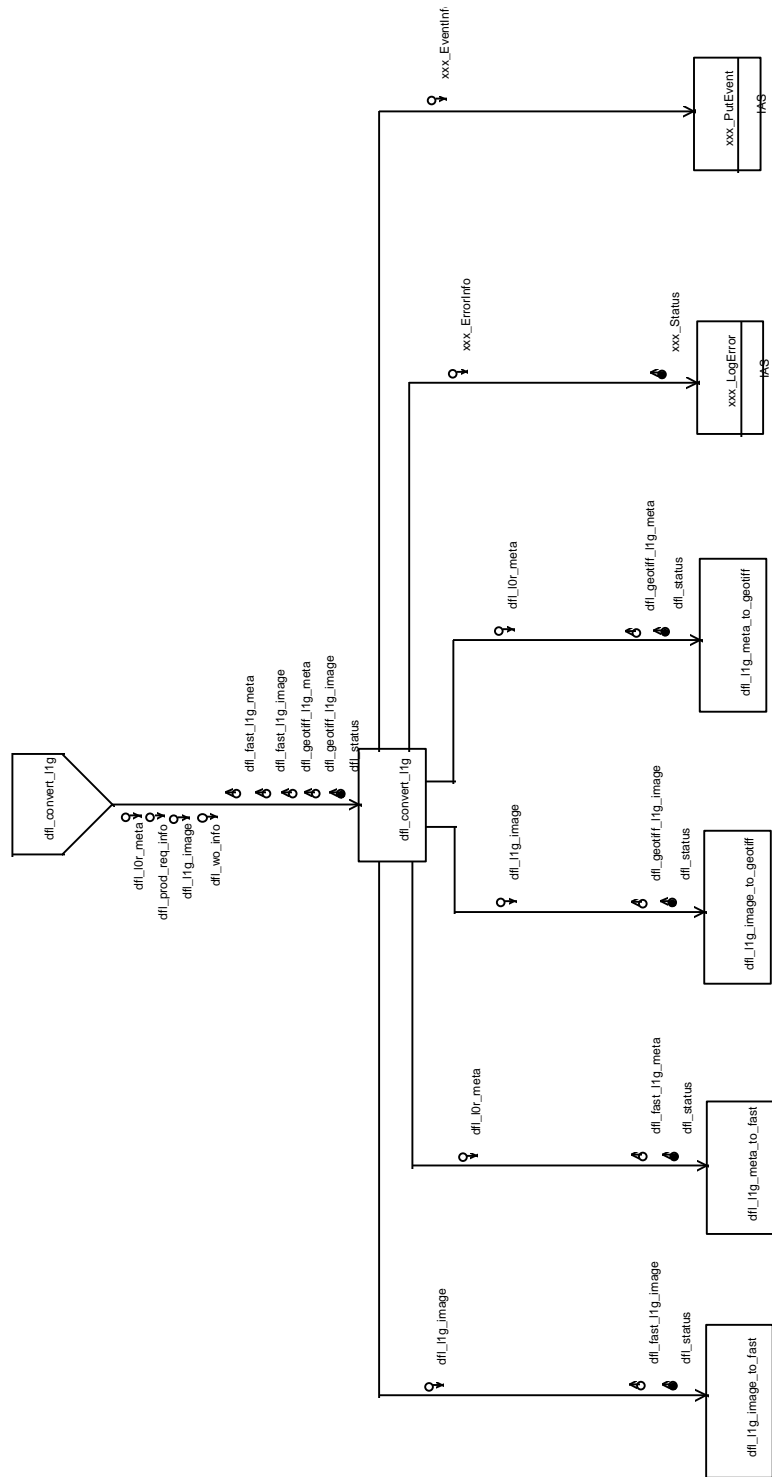


Figure 6-6. DFL Structure Chart (4 of 5)

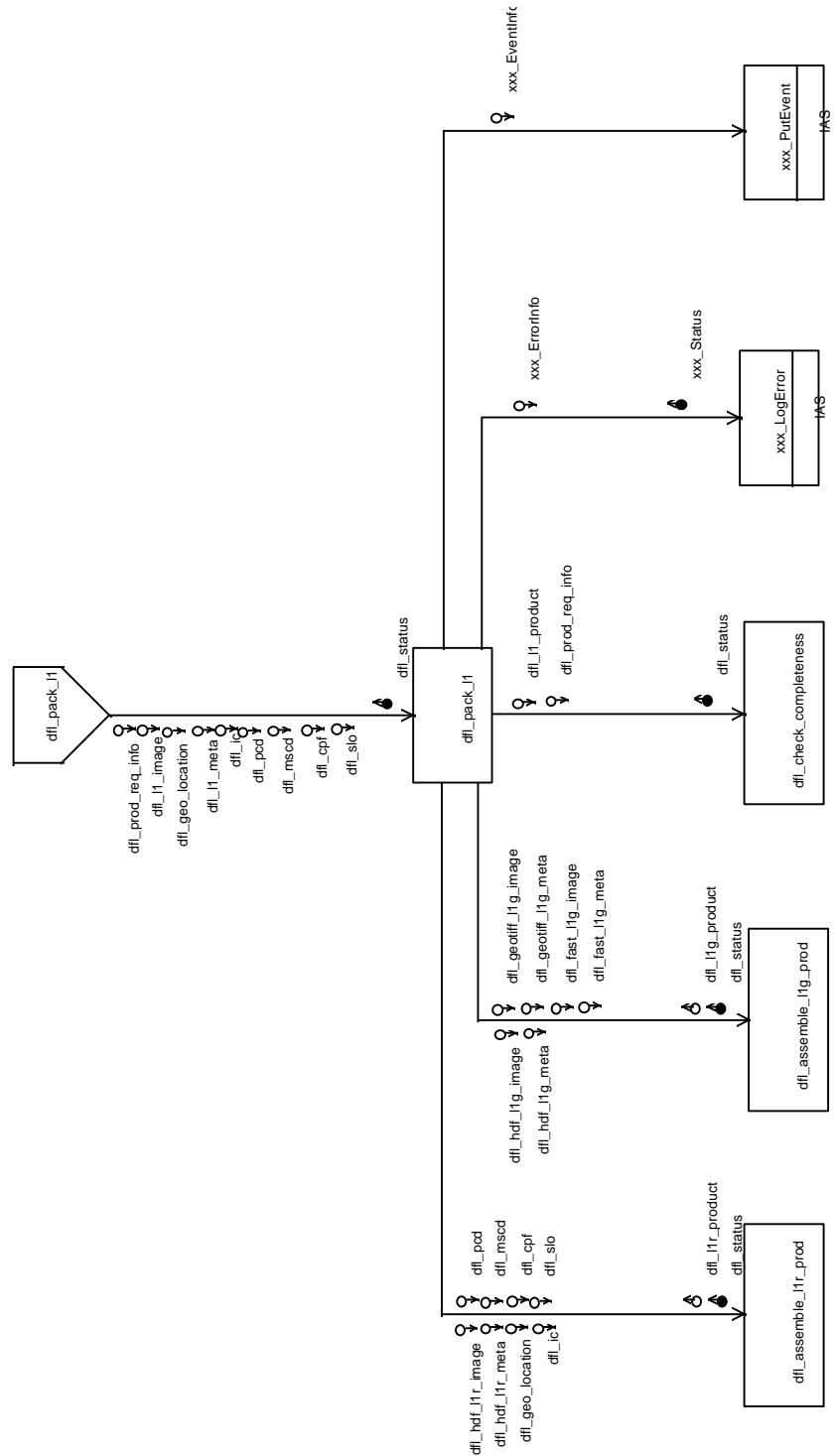


Figure 6-6. DFL Structure Chart (5 of 5)

## REVIEW

### **dfi\_assemble\_l1r\_prod—Assemble L1R Product**

#### **Parameters:**

dfi\_hdf\_l1r\_image : data\_in  
dfi\_pcd : data\_in  
dfi\_cpf : data\_in  
dfi\_l1r\_product : data\_out  
dfi\_status : control\_out  
dfi\_geo\_location : data\_in  
dfi\_slo : data\_in  
dfi\_mscd : data\_in  
dfi\_ic : data\_in  
dfi\_hdf\_l1r\_meta : data\_in

**Body:** dfi\_assemble\_l1r\_prod packs the final L1R product together. It retrieves the components of the L1R product based on the product request specification, labels each file according to naming convention and moves (or copies) them into one staging area (directory) for delivery. Generally, L1R product should include L1R image, a consensus PCD and a consensus MSCD, SLO, IC, CPF, geo-location table and meta data file. Only HDF format is available for L1R.

### **dfi\_check\_completeness—Check L1 Product Completeness**

#### **Parameters:**

dfi\_l1\_product : data\_in  
dfi\_status : control\_out  
dfi\_prod\_req\_info : data\_in

**Body:** dfi\_check\_completeness does the head counts for the final L1 product to match the original product request specifications. It validates all file names, checks data integrity and data consistency between the meta data file and L1 product components.

### **dfi\_convert\_l1g—Convert L1G to FAST or GeoTIFF Driver Module**

#### **Parameters:**

dfi\_fast\_l1g\_image : data\_out  
dfi\_fast\_l1g\_meta : data\_out  
dfi\_geotiff\_l1g\_image : data\_out  
dfi\_geotiff\_l1g\_meta : data\_out

## REVIEW

dfi\_status : control\_out

dfi\_l1g\_image : data\_in

dfi\_l0r\_meta : data\_in

dfi\_wo\_info : data\_in

dfi\_prod\_req\_info : data\_in

**Body:** dfi\_convert\_l1g produces L1G output image and meta data to GeoTIFF or FAST-C format as per the product request specification using COTS products and tool kits. In case of errors, event/error log and status notifications are posted before aborting.

**NOTE:** Input parameters from dfi\_main indicates the type of output format, plus the locations of L1G image data and meta data.

### PDL:

```
IF GeoTIFF format THEN
    CALL dfi_l1g_image_to_geotiff to produce L1G output image data in GeoTIFF
    format
    CALL dfi_l1g_meta_to_geotiff to construct L1G meta data, and append to L0R
    meta data, in GeoTIFF format
    IF error return THEN
        CALL xxx_LogError to write to error log
        CALL xxx_PutEvent to post event to database
        RETURN error dfi_status
    ENDIF
ELSE
    IF Fast-C format THEN
        CALL dfi_l1g_image_to_fast to produce L1G output image data in FAST-C
        format
        CALL dfi_l1g_meta_to_fast to construct L1G meta data, and append to L0R
        meta data, in FAST-C format
        IF error return THEN
            CALL xxx_LogError to write to error log
            CALL xxx_PutEvent to post event to database
            RETURN error dfi_status
        ENDIF
    ELSE
        CALL xxx_LogError to write to error log
        CALL xxx_PutEvent to post event to database
        RETURN error dfi_status
    ENDIF
ENDIF
RETURN with success
```

## REVIEW

### dfi\_format\_l1—Format L1R/L1G In HDF Driver Module

#### Parameters:

dfi\_hdf\_l1g\_image : data\_out  
dfi\_hdf\_l1\_meta : data\_out  
dfi\_status : control\_out  
dfi\_l1r\_or\_l1g\_image : data\_in  
dfi\_l0r\_meta : data\_in  
dfi\_wo\_info : data\_in  
dfi\_prod\_req\_info : data\_in  
dfi\_hdf\_l1r\_image : data\_out

**Body:** dfi\_format\_l1 reads L1R output from RPS or L1G output from GPS, and uses COTS products to make a HDF-EOS output. It also constructs L1 product meta data file in HDF format and append to the L0R meta data.

**NOTE:** Input parameters from dfi\_main indicates the type of L1 product, the type of output format, and the locations of all related L1 files.

#### PDL:

```
IF L1R THEN
  IF HDF-EOS THEN
    CALL dfi_l1r_image_to_hdf to read L1R image data, dfi_l1r_image, and pack
      it to HDF format, dfi_hdf_l1r_image
    CALL dfi_meta_to_hdf to construct L1R meta data file and append to L0R meta
      data in HDF format
  ELSE
    CALL xxx_LogError to write to error log
    CALL xxx_PutEvent to post event to database
    RETURN error dfi_status
  ENDIF
ELSE
  IF L1G and HDF-EOS THEN
    CALL dfi_l1g_image_to_hdf to read L1G image data and pack it to HDF
      format, dfi_hdf_l1g_image
    CALL dfi_meta_to_hdf to construct L1G meta data file and append to LPS meta
      data file in HDF format
  ELSE
    CALL xxx_LogError to write to error log
    CALL xxx_PutEvent to post event to database
    RETURN error dfi_status
  ENDIF
```

## REVIEW

```
ENDIF  
RETURN with success
```

### **dfi\_init—DFL-Initialization Driver Module**

#### **Parameters:**

```
dfi_status : control_out  
dfi_prod_req_info : data_out  
dfi_wo_info : data_out  
xxx_odl_filename : data_in
```

**Body:** dfi\_init initializes the DFL task. It calls xxx\_GetParam to process the input script parameters, xxx\_odl\_filename, which leads to an ODL file containing product format request information, such as the locations of all associated intermediate L1 files. It then connects to the LPGS database. If information other than that provided by the ODL file is needed, dfi\_init retrieves them from the database.

#### **PDL:**

```
CALL xxx_ConnectToDB to access the LPGS database  
CALL xxx_GetParam to access the ODL file to process xxx_odl_filename and to obtain  
dfi_prod_req_info and/or dfi_wo_info, for instance, the files locations associated with  
specific product request  
IF error occurs THEN  
    CALL xxx_LogError to report to error log  
    CALL xxx_PutEvent to post event to database  
    CALL xxx_DisconnectFromDB to disconnect the LPGS database  
    RETURN error dfi_status  
ENDIF  
CALL xdb_get_task_config to retrieve additional configuration parameters, that are not  
provided by the ODL file  
IF database access error THEN  
    CALL xxx_LogError to write to error log  
    CALL xxx_PutEvent to post event to database  
    CALL xxx_DisconnectFromDB to disconnect the LPGS database  
    RETURN error dfi_status  
ENDIF  
RETURN with success
```

## REVIEW

### **df\_l1g\_image\_to\_fast—Format L1G Image In FAST-C**

#### **Parameters:**

df\_l\_status : control\_out

df\_l\_fast\_l1g\_image : data\_out

df\_l\_l1g\_image : data\_in

**Body:** df\_l\_l1g\_image\_to\_fast reads L1G output from GPS (df\_l\_l1g\_image), and packs into FAST-C format using COTS programs.

### **df\_l1g\_image\_to\_geotiff—Format L1G Image in GeoTIFF**

#### **Parameters:**

df\_l\_status : control\_out

df\_l\_geotiff\_l1g\_image : data\_out

df\_l\_l1g\_image : data\_in

**Body:** df\_l\_l1g\_image\_to\_geotiff reads L1G output from GPS (df\_l\_l1g\_image), and packs into GeoTIFF format using COTS programs.

### **df\_l1g\_image\_to\_hdf—Format L1G Image In HDF**

#### **Parameters:**

df\_l\_l1g\_image : data\_in

df\_l\_hdf\_l1g\_image : data\_out

df\_l\_status : control\_out

**Body:** df\_l\_l1g\_image\_to\_hdf reads L1G output from GPS (df\_l\_l1g\_image) and packs into HDF format using COTS programs.

### **df\_l1g\_meta\_to\_fast—Format L1G Meta Data In FAST-C**

#### **Parameters:**

df\_l\_status : control\_out

df\_l\_fast\_l1g\_meta : data\_out

df\_l\_l0r\_meta : data\_in

**Body:** df\_l\_meta\_to\_fast reads L0R meta data (df\_l\_l0r\_meta) from product request input directory, and appends to the LPS and ECS sections with a LPGS meta data section. The LPGS meta data reflects characteristics of L1G product and its quality evaluations. The whole meta data is then converted to FAST-C format using COTS programs.

## REVIEW

### **df\_l1g\_meta\_to\_geotiff—Format L1G Meta Data in GeoTIFF**

#### **Parameters:**

df\_l\_status : control\_out  
df\_l\_geotiff\_l1g\_meta : data\_out  
df\_l\_l0r\_meta : data\_in

**Body:** df\_l\_meta\_to\_geotiff reads L0R meta data (df\_l\_l0r\_meta) from product request input directory, and appends to the LPS and ECS sections with a LPS meta data section. The LPS meta data reflects characteristics of L1G product and its quality evaluations. The whole meta data is then converted to GeoTIFF format using COTS programs.

### **df\_l1r\_image\_to\_hdf—Format L1R Image in HDF**

#### **Parameters:**

df\_l1r\_image : data\_in  
df\_l\_hdf\_l1r\_image : data\_out  
df\_l\_status : control\_out

**Body:** df\_l1r\_image\_to\_hdf reads RPS output, df\_l1r\_image, from appropriate product request intermediate directory, and packs it to HDF format using COTS programs.

### **df\_l\_main—DFL Driver Module**

#### **Parameters:** None

**Body:** df\_l\_main controls the process sequence of DMS Format L1-Product (DFL). It is a script initiated by PCS Work Order Controller. df\_l\_main first calls df\_l\_init to initialize the DFL task. df\_l\_init processes input script parameters, xxx\_odl\_filename, or accesses the LPS database to get all related L1 product request information, such as the locations of all L1 product files. After successfully locating the image and all non-image files, df\_l\_main proceeds to format the L1 product based on the PCS input requests. If L1R product is requested, df\_l\_main calls df\_l\_format\_l1 to make L1R product in HDF-EOS format. If L1G product is requested, it must be presented in one of the three formats: HDF-EOS, FAST-C or GeoTIFF. Therefore, df\_l\_main calls df\_l\_format\_l1 for HDF-EOS output, and calls df\_l\_convert\_L1G for GeoTIFF or FAST-C output. Finally, df\_l\_main calls df\_l\_pack\_l1 to package the L1 image and all requested non-image data into a L1 product. The product is verified for completeness against the original L1 product request before being moved to a stage directory for ECS delivery. df\_l\_main then calls xxx\_DisconnectFromDB to disconnect the database and sends processing status to the PCS Controller.

#### **PDL:**

CALL df\_l\_init to initialize DFL task, to get product request and work order  
configurations from the ODL file, such as the locations of all L1 files  
IF error df\_l\_status THEN

## REVIEW

```
        SET error status
        GOTO EXIT
    ENDIF
    IF L1R product request THEN
        CALL dfl_format_l1 to produce L1R image and non-image files in HDF-EOS
        format
    ENDIF
    IF error dfl_status THEN
        SET error status
        GOTO EXIT
    ENDIF
    IF L1G product request THEN
        DOCASE l1g_format
            CASE HDF-EOS
                CALL dfl_format_l1 to produce L1G image and metadata in HDF-EOS
                format
            CASE GeoTIFF
                CALL dfl_convert_l1g to produce L1G image and metadata in GeoTIFF
                format
            CASE FAST-C
                CALL dfl_convert_l1g to produce L1G image and metadata in FAST-C
                format
        ENDDO
    ENDIF
    IF error dfl_status THEN
        SET error status
        GOTO EXIT
    ENDIF
    CALL dfl_pack_l1 to package L1 product, check the completeness and stage it to a
    product delivery directory
    IF error dfl_status THEN
        SET error status
    ENDIF
    EXIT:
        CALL xxx_DisconnectFromDB to disconnect the LPGS database
    RETURN
```

### **dfl\_meta\_to\_hdf—Format Meta Data In HDF**

#### **Parameters:**

dfl\_l0r\_meta : data\_in  
dfl\_prod\_req\_info : data\_in

## REVIEW

df\_l\_status : control\_out

df\_l\_hdf\_l1\_meta : data\_out

**Body:** df\_l\_meta\_to\_hdf reads L0R meta data (df\_l0r\_meta) from appropriate product request input directory, and appends to the LPS and ECS sections with a LPGS meta data section. The LPGS meta data reflects characteristics of L1R or L1G product and its quality evaluations. The whole meta data is written in HDF format using COTS programs.

### df\_l\_pack\_l1—Pack L1 Product Driver Module

#### Parameters:

df\_l\_status : control\_out

df\_l\_l1\_image : data\_in

df\_l\_geo\_location : data\_in

df\_l\_l1\_meta : data\_in

df\_l\_pcd : data\_in

df\_l\_mscd : data\_in

df\_l\_cpf : data\_in

df\_l\_slo : data\_in

df\_l\_ic : data\_in

df\_l\_prod\_req\_info : data\_in

**Body:** df\_l\_pack\_l1 locates the L1 image and all associated non-image data from appropriate product request intermediate directories. The required ancillary files for L1R product include a consensus PCD data, a consensus MSCD data, Calibration Parameter File (CPF), geographic location table, internal calibrator data, and scan line offset data. Both L1R and L1G product consist of a metadata file, a representative description of the product, based on the nature of the product request, the L1 and L0 image characteristics, and the accrued quality evaluation during the L1 processing. The locations of all L1 product files are stored in the LPGS database. df\_l\_pack\_l1 then packs the L1 product with appropriate file labeling, and moves them to a designated directory for delivery. Finally, df\_l\_pack\_l1 checks the completeness of the L1 product against the product request specification. If error occurs, event/error log and status notifications are posted before aborting.

**NOTE:** Input parameters from df\_l\_main indicates the type of L1 product, and the locations of all component files.

#### PDL:

IF L1R THEN

CALL df\_l\_assemble\_l1r-prod to access L1R image, PCD, MSCD, meta, geo-location, SLO, IC, CPF from intermediate directories in HDF format, label them

## REVIEW

```
following LPGS naming convention and move them to a delivery staging
directory for ECS to “ftp-get”
ELSE
  IF L1G THEN
    CALL dfl_assemble_l1g_prod to access L1G image and meta data from
    intermediate directories with appropriate format, label them following LPGS
    naming convention and move them to a delivery staging directory for ECS
    to “ftp-get”
  ENDIF
ENDIF
CALL dfl_check_completeness to check for completeness of the L1 product against the
specification of product request
IF error return in any of above steps THEN
  CALL xxx_LogError to write to error log
  CALL xxx_PutEvent to post event in database
  RETURN error dfl_status
ENDIF
RETURN
```

### 6.3.5 DMS Xmit L1 Product (DXL) Task

This subsection describes the DXL task software.

#### 6.3.5.1 Task Overview

This task is responsible for the transmission of the finished L1R/L1G products to ECS. The task is started by the LPGS Software Initialization task and runs continuously. This task polls the database for L1 products that are ready for transfer to ECS. DXL places a Product Delivery Record (PDR) file in a predefined directory on the LPGS for ECS access. The task polls this directory for responses from ECS which are either Product Delivery Record Discrepancy (PDRD) files or Product Acceptance Notification (PAN) files.

#### 6.3.5.2 Initialization

The task is started by the LPGS Initialization Task and accesses initialization data from the database for configuration.

#### 6.3.5.3 Normal Operation

The DXL task periodically polls the database for L1 products ready for transfer to ECS. When a ready product is found, a PDR is placed in the LPGS directory specified for ECS access. DXL polls this directory for response files from ECS. The response can be either a PDRD signifying an error in the PDR or a PAN signifying success or failure in transferring the L1 product files. If the transfer was successful, the database is updated to indicate the successful transfer and to indicate the image files can be deleted. If the failure was in the PDR, DXL regenerates the PDR. If the error was with the image files an error is reported back to the operator. In all cases, DXL deletes the protocol files (PDR, PAN and PDRD).

## REVIEW

### 6.3.5.4 Error Handling

The DXL task establishes a signal handler to capture all fatal UNIX signals which would cause the task to abort. The signal handler sends an error message to the operator, disconnects from the database, and gracefully shuts down the task. Nonfatal errors are logged to the LPGS error log and an error notification is sent to the user interface.

### 6.3.5.5 Design

This subsection presents the design of the DXL task. The structure chart for the task is illustrated in Figure 6-7. The module specifications for the DXL task are provided (in alphabetical order) below.

#### **dxl\_check\_responses—Check For ECS Responses**

##### **Parameters:**

dxl\_status : control\_out

**Body:** This module checks for and processes protocol responses for the PDR files placed by dxl\_prepare\_pdr. It calls dxl\_poll\_for\_response to get a list of new files in the predefined directory. dxl\_read\_response is called to interpret the responses. If a PDRD file was read, the status return indicates whether the PDR needs to be resent (by a call to dxl\_prepare\_pdr) or appropriate status returned to operator by xxx\_PutEvent.

#### **dxl\_create\_pdr—Create The Product Delivery Request File**

##### **Parameters:**

xxx\_prod\_req\_id : data\_in

dxl\_status : control\_out

**Body:** This module creates the Product Delivery Request (PDR) files for the specified product generation request. This module calls xdb\_get\_prod\_req to get the product request information from the database. Then, this module creates (open, write, & close) the L1\_Prod\_Avail\_Notice PDR file. The file is placed in the predefined directory accessed by ECS. Finally, this module updates the state of the product request to indicate the PDR file has been created using xdb\_update\_prod\_req\_state.

#### **dxl\_delete\_protocol\_files—Delete Protocol Files**

##### **Parameters:**

dxl\_resp\_file\_id : data\_in

dxl\_status : control\_out

**Body:** This unit deletes the PDR, PDRD, and PAN files associated with a L1 product from the predefined protocol directory.

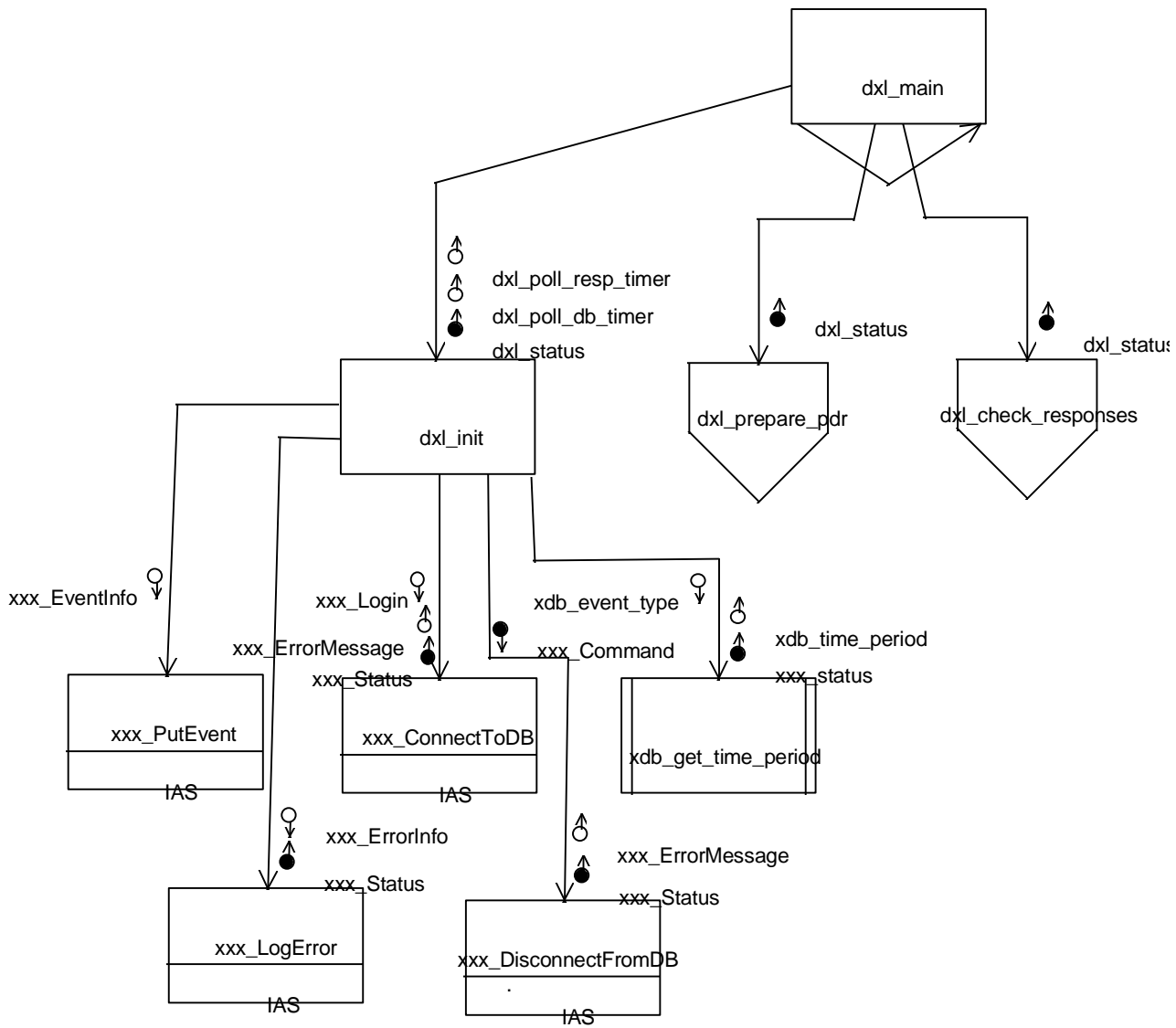


Figure 6-7. DXL Structure Chart (1 of 4)

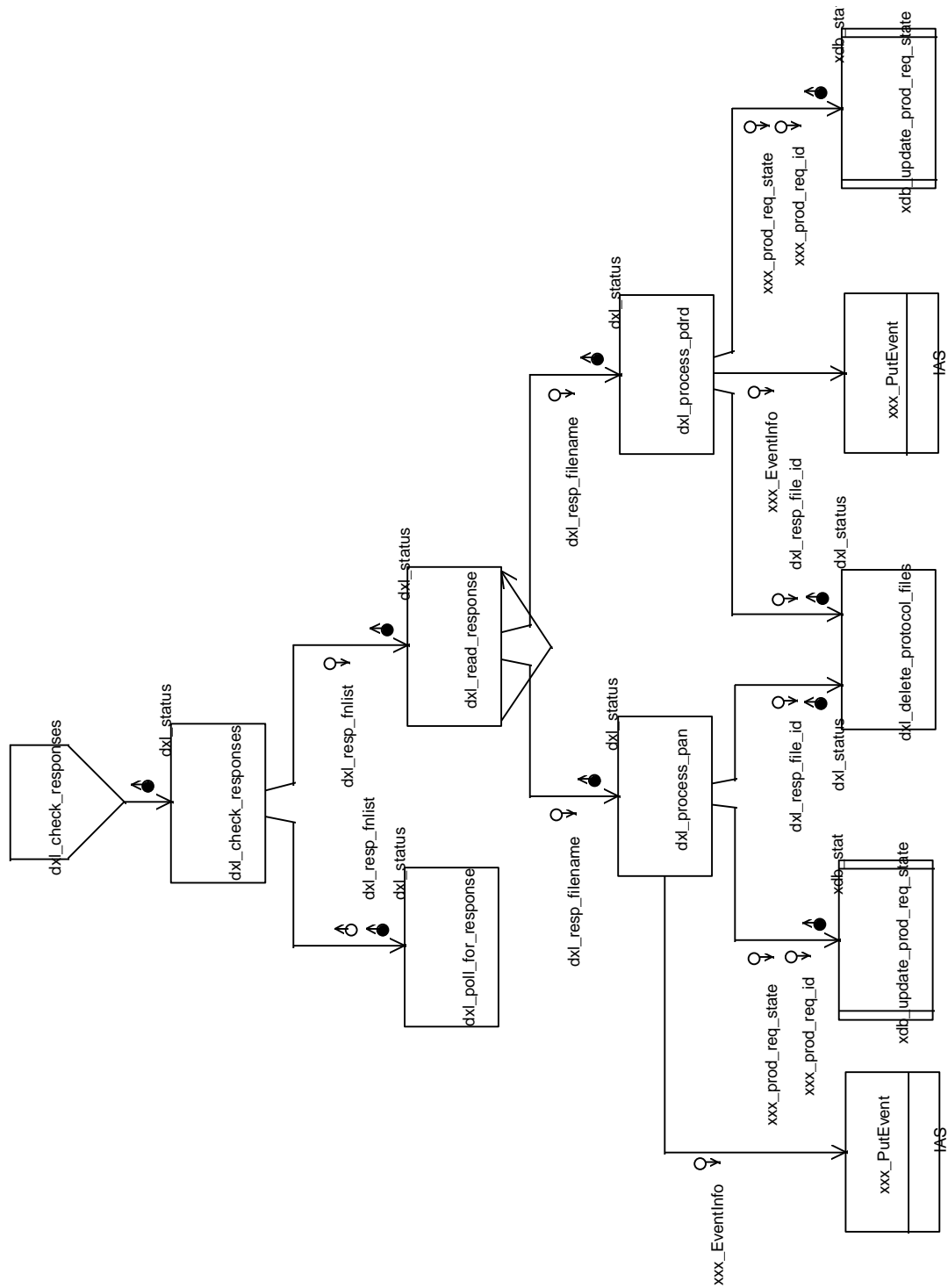


Figure 6-7. DXL Structure Chart (2 of 4)

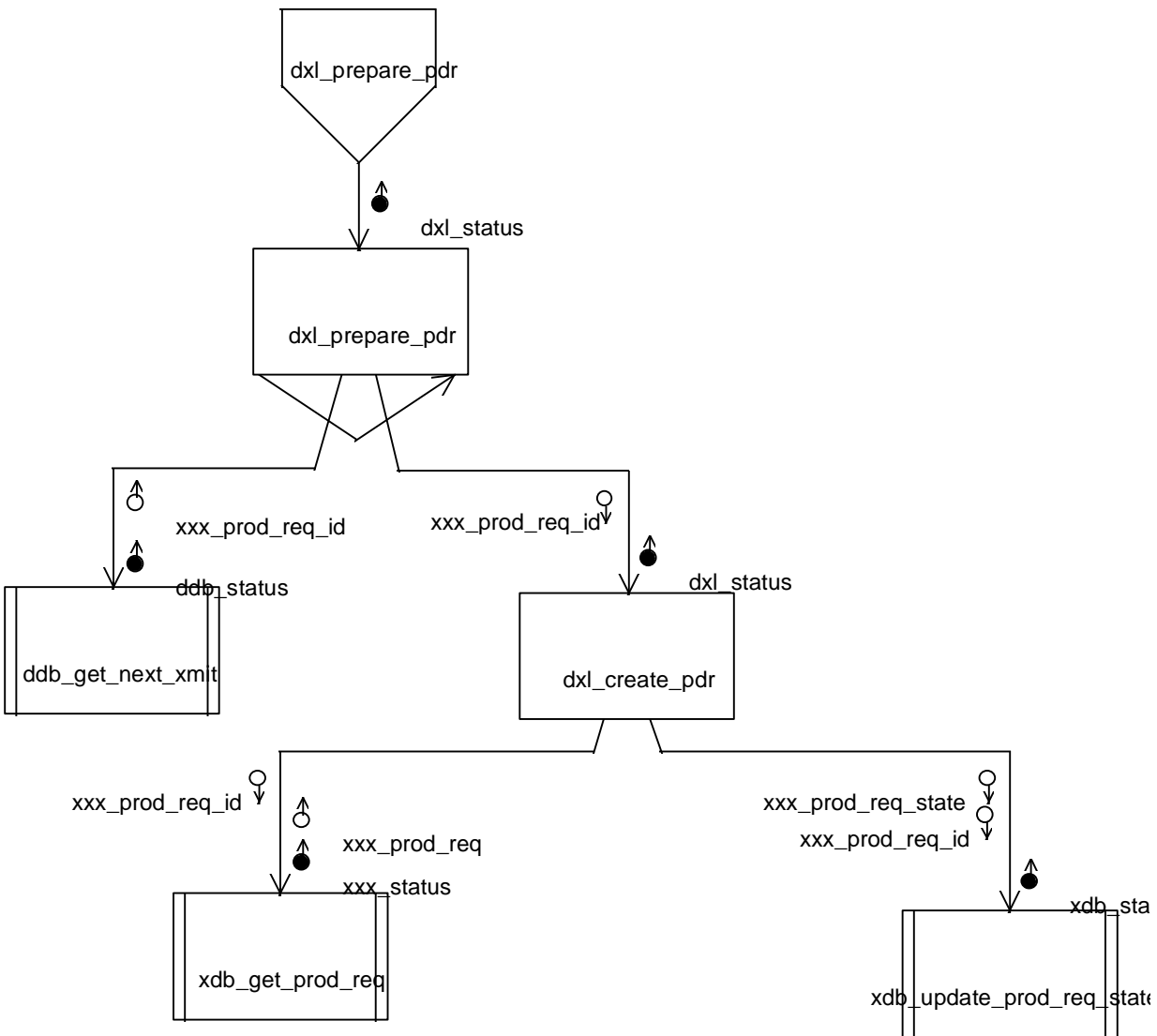


Figure 6-7. DXL Structure Chart (3 of 4)

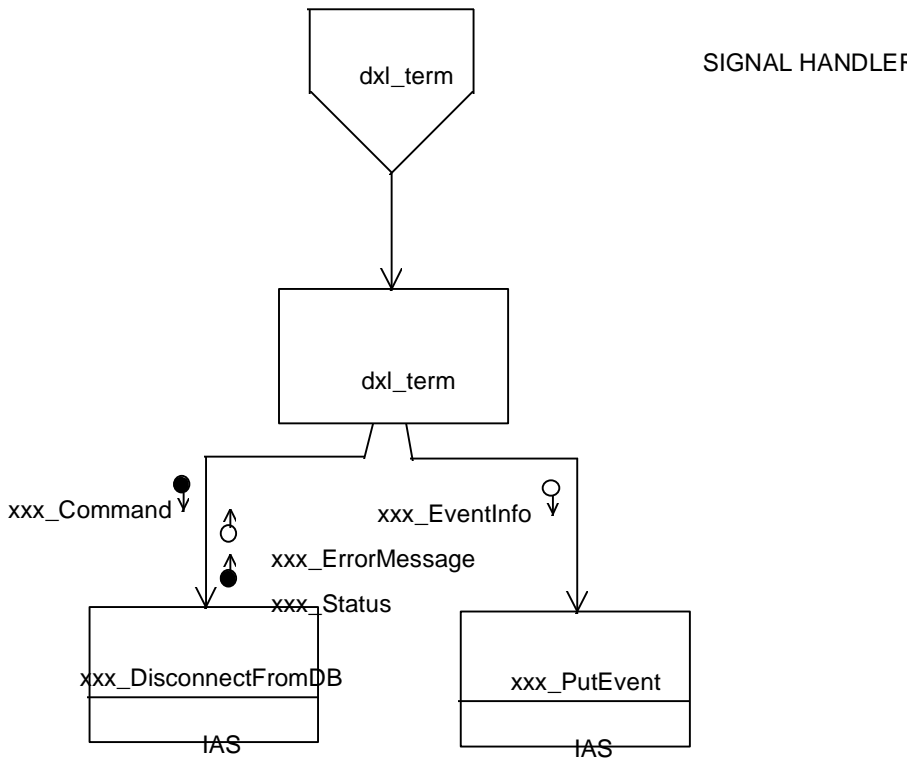


Figure 6-7. Dxl Structure Chart (4 of 4)

## REVIEW

### **dxl\_init—DXL Task Initialization**

#### **Parameters:**

dxl\_poll\_resp\_timer : data\_out

dxl\_poll\_db\_timer : data\_out

dxl\_status : control\_out

**Body:** The dxl\_init module calls xxx\_ConnectToDB to connect to the database and sets up a signal handler for process termination. Then, this module calls xdb\_get\_time\_period to get the time period for checking the database for product requests that need a PDR file created. This module then calls xdb\_get\_time\_period a second time to get the time period for polling for L1 Product Available Response (PDRD or PAN) files from ECS. This module returns appropriate status if an error occurs during processing. Modules xxx\_LogError and xxx\_DisconnectFromDB are called if an error occurs.

### **dxl\_main—DXL Main Process**

#### **Parameters:** None

**Body:** This module is responsible for managing the transfer of L1 products to ECS. The transfer is accomplished by placing a PDR file in a predefined directory on the LPGS system for access by ECS. After accessing this file, and the L1 product if no errors were encountered in the PDR, ECS places a response file in the same directory as the PDR. This response file is either a PAN or a PDRD. If the transfer was successful, the files associated with the Product Request are marked for deletion. If unsuccessful, the PDR is reproduced if ECS indicated a problem with the PDR, otherwise an alert is sent to the operator. In either case, the protocol files (PAN, PDR, PDRD) are deleted. The database and response file directory are polled at time increments specified in the configuration file. The module first calls dxl\_init to connect to the database and acquire configuration information including timer increments. It then calls both dxl\_prepare\_pdr and dxl\_check\_responses periodically to determine if there are any L1 products to transmit or responses for PDR's placed for ECS access.

#### **PDL:**

```
CALL dxl_init to connect to database and get configuration items including timer
increments
IF error reported by dxl_init THEN
  EXIT
ENDIF
DO FOREVER
  SET the timer to expire for the next event
  WAIT for the timer to expire
  IF it is time to check for L1 products ready to send THEN
    CALL dxl_prepare_pdr to process ready products
  ELSE
    IF it is time to poll the protocol directories THEN
```

## REVIEW

```
        CALL dxl_check_responses to poll the directory for messages
    ENDIF
ENDIF
ENDDO
EXIT
```

### **dxl\_poll\_for\_response—Poll Protocol Directory For Response Files**

#### **Parameters:**

dxl\_status : control\_out  
dxl\_resp\_fnlist : data\_out

**Body:** This module uses the EgFTP COTS product to query a predefined directory for entries, and builds a list of new entries.

### **dxl\_prepare\_pdr—Prepare The L1 Product Available Notice PDR Files**

#### **Parameters:**

dxl\_status : control\_out

**Body:** This module is called periodically to create the Product Delivery Request (PDR) files for the product generation requests. After the Level 1 product has been generated, the status for the product request is updated to indicate the product is ready for delivery. When this module is called, it determines all of the product requests that are ready for delivery. This module calls ddb\_get\_next\_xmit to determine the next product request that needs to be transmitted. Then, this module calls dxl\_create\_pdr to create the L1\_Prod\_Avail\_Notice PDR file.

#### **PDL:**

```
Set done to FALSE
DO WHILE not done
    CALL ddb_get_next_xmit to get the next product request that needs a PDR file
    created
    IF there is a product request that needs a PDR created THEN
        CALL dxl_create_pdr to create the PDR file for the product request
    ELSE
        Set done to TRUE
    ENDIF
ENDDO
```

### **dxl\_process\_pan—Process ECS PAN File**

#### **Parameters:**

dxl\_status : control\_out  
dxl\_resp\_filename : data\_in

## REVIEW

**Body:** This module reads a PAN file named as input and extracts the product request id and protocol file ids. This information is used in updating product status (xdb\_update\_prod\_req\_state) if the PAN indicates successful result. If the PAN indicates failure, the specific error from the PAN is recorded using xxx\_PutEvent. Protocol files are deleted by dxl\_delete\_protocol\_files.

### **dxl\_process\_pdrd—Process ECS PDRD File**

**Parameters:**

dxl\_resp\_filename : data\_in

dxl\_status : control\_out

**Body:** This module reads a PDRD file named as input. It interprets the contents to return status indicating the source of the error. It logs the event with xxx\_PutEvent and calls dxl\_delete\_protocol\_files to remove the PDR and PDRD.

### **dxl\_read\_response—Read ECS Response File**

**Parameters:**

dxl\_status : control\_out

dxl\_resp\_fnlist : data\_in

**Body:** This module reads and interprets each file in dxl\_resp\_fnlist. If the file type is PAN it calls dxl\_process\_pan to complete processing, if it is a PDRD dxl\_process\_pdrd is called.

### **dxl\_term—DXL Task Termination Signal Handler**

**Parameters:** None

**Body:** This module terminates the DXL process. This module disconnects from the database. Then, this module terminates the DXL process.

## **6.3.6 DMS Resource Manager (DRM) Task**

This subsection describes the DRM task software.

### **6.3.6.1 Task Overview**

The DRM task manages the LPGS product request file system. The DRM task is started by the PSI task and runs as a background task. This task periodically deletes all product request files and directories that have been marked for deletion. In addition, this task can receive operator commands to delete any product request or any work order files. After each deletion, DRM updates the database to indicate that the files and directories are no longer available on the LPGS system. The DRM task also monitors the disk usage, and alerts the operator when disk usage exceeds the operator configurable limits.

## REVIEW

### 6.3.6.2 Initialization

The DRM task is started by the PSI task. The DRM initialization includes connecting to the database, creating a signal handler, and retrieving configuration information from the database. Any errors encountered during initialization results in the DRM task exiting.

### 6.3.6.3 Normal Operation

The DRM task runs when the timer expires or when an operator request is received. When the timer expires, DRM cleans up the disk and checks the disk usage. The DRM task scans the database and determines which product request or work order files and directories have been marked for deletion. After deleting the files, the directories, and all subdirectories, DRM updates the database to indicate they were deleted. DRM also evaluates the disk utilization, and it alerts the operator if any disk usage exceeds the threshold. When an operator request is received, this task immediately deletes the file or directory, updates the product request state in the database, and sends a response to the operator.

### 6.3.6.4 Error Handling

The DRM task establishes a signal handler to capture all fatal UNIX signals which would cause the task to abort. The signal handler sends an error message to the operator, disconnects from the database, and gracefully shuts down the task. Nonfatal errors are logged to the LPGS event log and an error notification is sent to the database.

### 6.3.6.5 Design

This subsection presents the design of the DRM task. The structure chart for the software is illustrated in Figure 6-8. The module specifications for the DRM task are provided (in alphabetical order) below.

#### **drm\_chk\_disk—Check Disk Usage Status**

##### **Parameters:**

drm\_config\_info : data\_in

drm\_status : control\_out

drm\_disk\_use\_error : data\_out

**Body:** This module monitors the LPGS disk usage status by checking the amount of available disk space periodically. If the free disk space is below a specified operation threshold, this module returns a disk-usage-error message to drm\_chk\_disk\_usage.

##### **PDL:**

```
ISSUE UNIX command “df -iPI” to get disk and i-node usage status
REDIRECT the status output to a temporary working file, temp.dat
IF error in process THEN
    RETURN with error drm_status
```

REVIEW

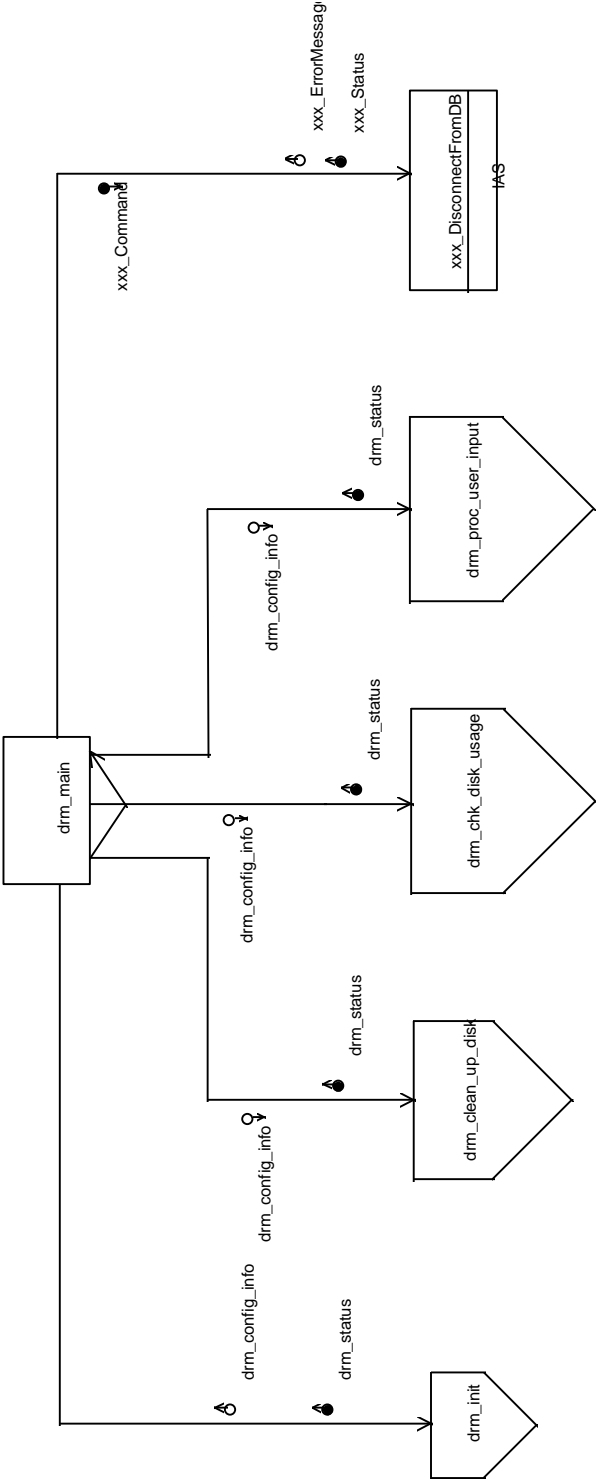


Figure 6-8. DRM Structure Chart (1 of 6)

REVIEW

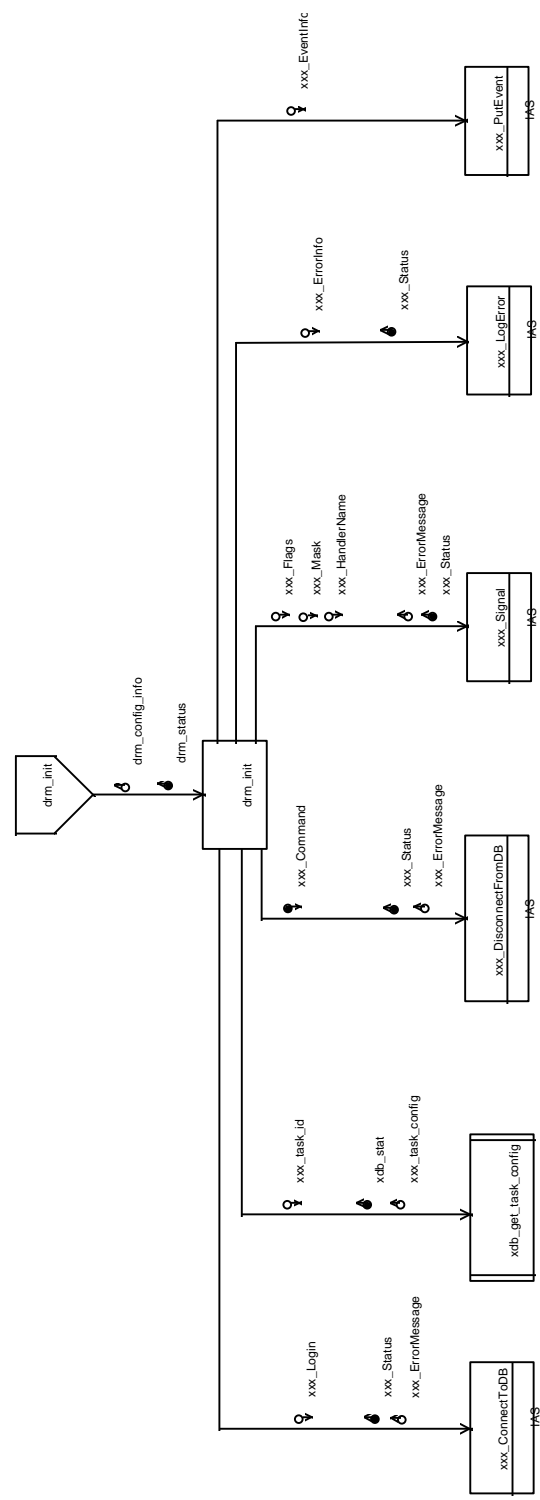


Figure 6-8. DRM Structure Chart (2 of 6)

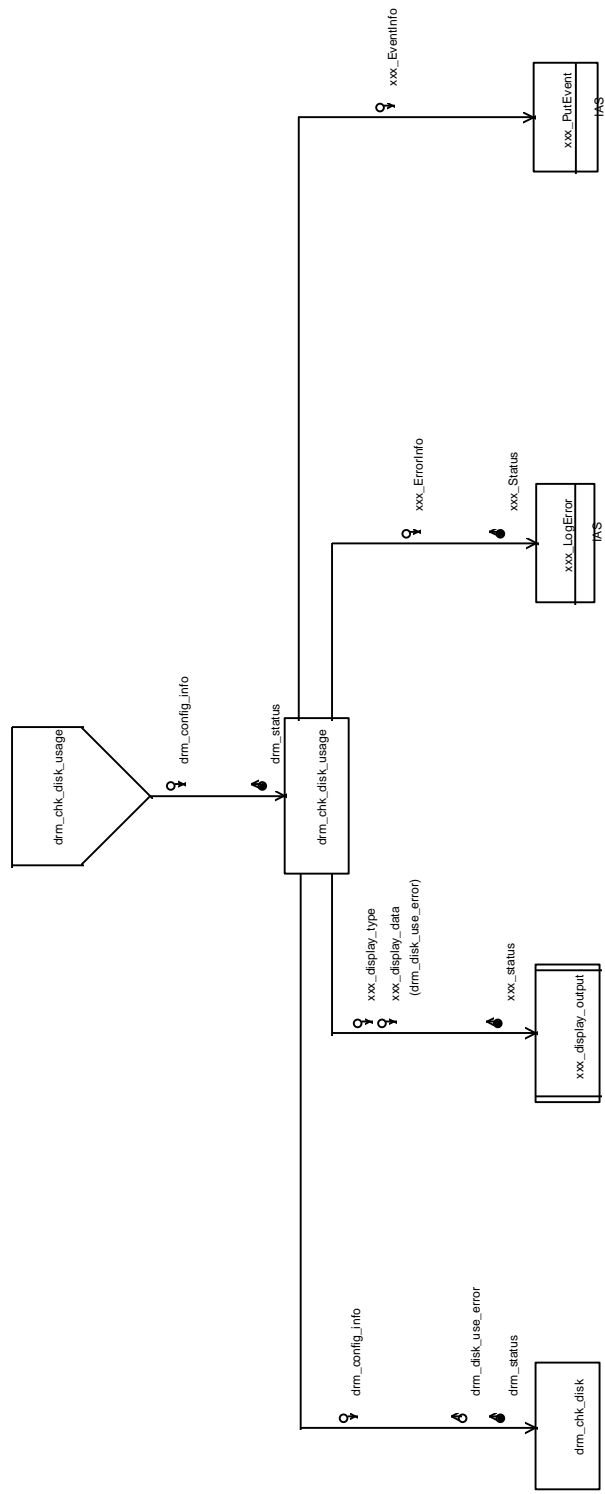
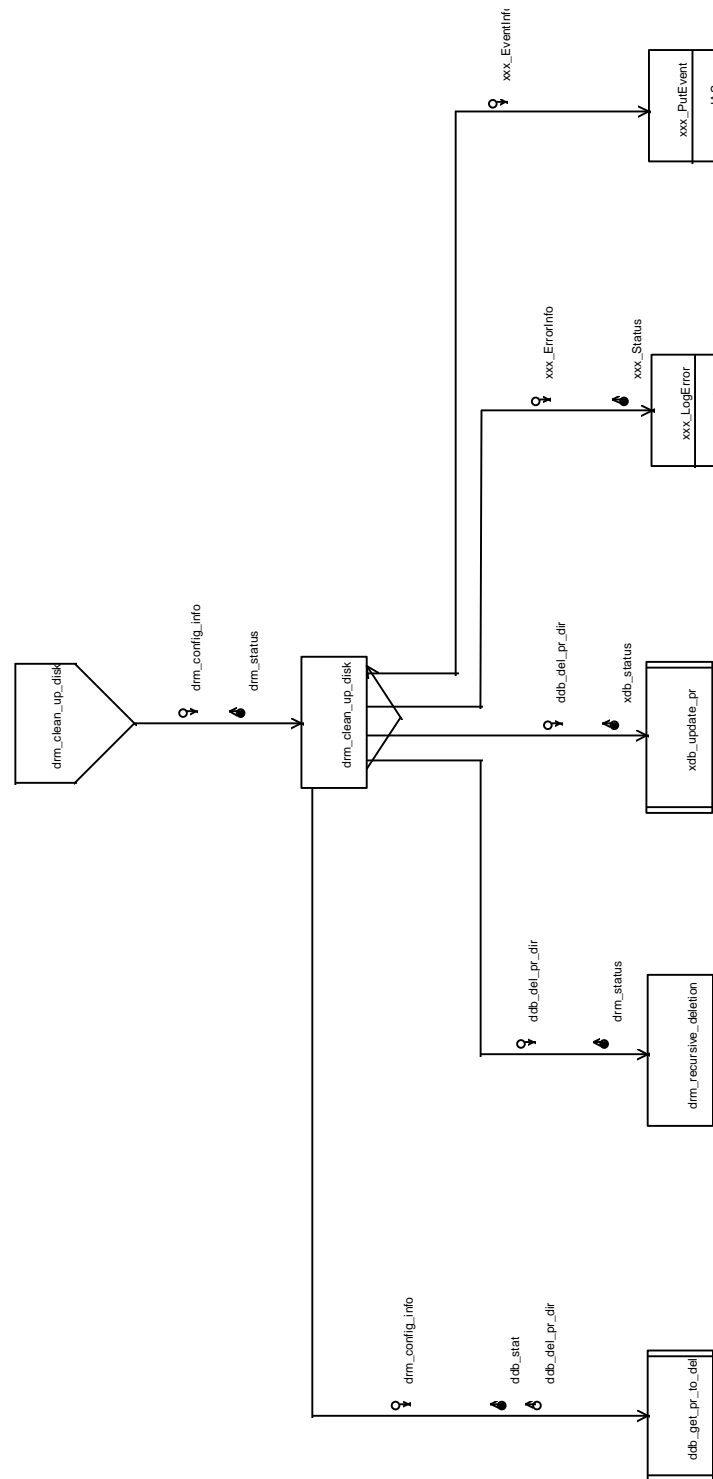


Figure 6-8. DRM Structure Chart (3 of 6)

## REVIEW



**Figure 6-8. DRM Structure Chart (4 of 6)**

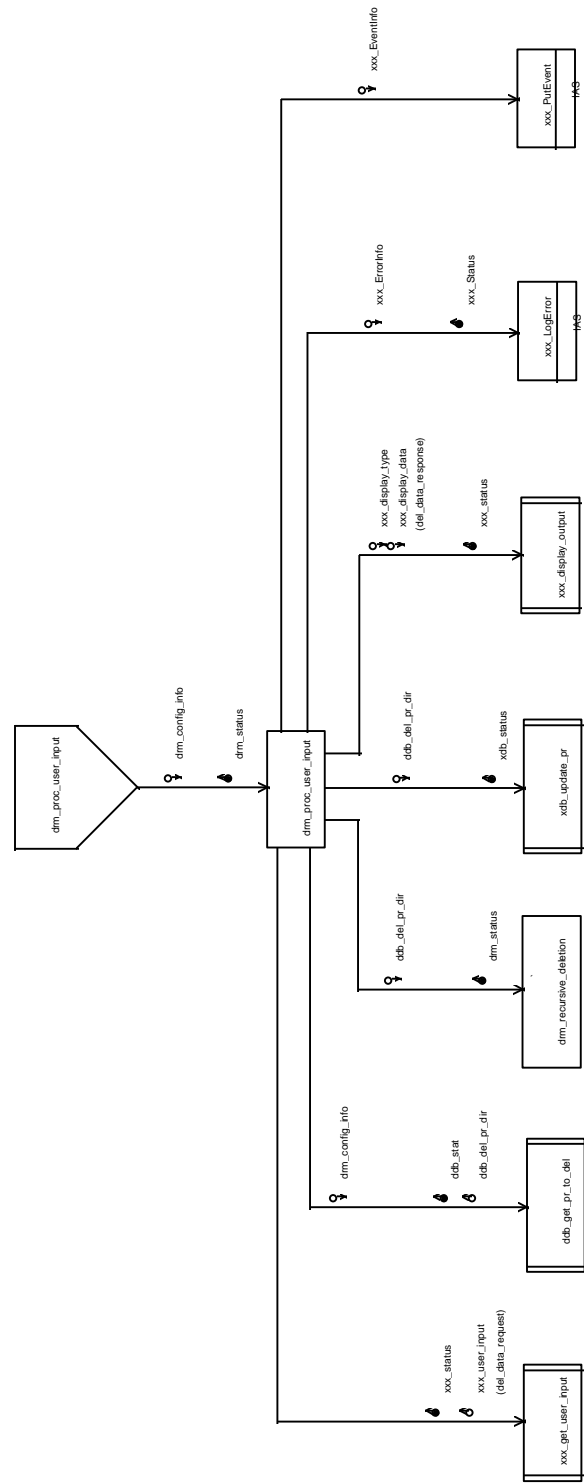


Figure 6-8. DRM Structure Chart (5 of 6)

REVIEW



Figure 6-8. DRM Structure Chart (6 of 6)

## REVIEW

```
ENDIF
DOUNTIL no more lines left
  READ one line from temp.dat
  FETCH space usage percentage for each RAID disk pack
  IF the use percentage is over a preset threshold, specified in "drm_config_info"
  THEN
    SET drm_disk_use_error flag
    EXIT the loop
  ENDIF
ENDDO
RETURN
```

### **drm\_chk\_disk\_usage—Check-Disk-Usage Driver Module**

#### **Parameters:**

drm\_status : control\_out  
drm\_config\_info : data\_in

**Body:** This module checks the amount of available disk space. If the amount of disk space available is less than a specified threshold, an error message is logged to the LPGS error log and an error notification is sent to the user interface.

#### **PDL:**

```
CALL drm_chk_disk to survey the current usage status of each LPGS system disk
IF error return THEN
  Call xxx_LogError to write to error log
  CALL drm_PutEvent to post event to database
  RETURN with error drm_status
ELSE
  IF any disk usage below threshold THEN
    CALL xxx_display_output to send warning message to operator
    Call xxx_LogError to send warning message to system error log
    CALL drm_PutEvent to post event to database
    RETURN with error drm_status
  ENDIF
ENDIF
RETURN
```

### **drm\_clean\_up\_disk—Clean-Up-Disk Driver Module**

#### **Parameters:**

drm\_config\_info : data\_in  
drm\_status : control\_out

## REVIEW

**Body:** This module queries the LPGS database to determine which product request data and/or directories are marked for deletion. This includes all files in the product request input directory and all associated work order intermediate directories, plus the L1 product files in the product delivery directory. It then deletes all files in those directories plus their subdirectories. Finally, it updates the LPGS database to indicate this change.

### PDL:

```
CALL ddb_get_pr_to_del to retrieve a list of product requests that are marked for
deletion, using drm_config_info as a guideline. Locate all associated data
directories that are marked for deletion.
IF deletion directories are found THEN
  DOFOR each product request in the list
    CALL drm_recursive_deletion to clean up all files in the input and intermediate
    directories and all its sub-directories, plus the L1 product in the product
    delivery directory
  IF error encountered THEN
    CALL xxx_LogError to write to error log
    CALL drm_PutEvent to post event to database
    SET error drm_status
  ELSE
    CALL xdb_update_pr to update the product request and/or work order
    entities in the database, reflecting that all input, intermediate, and output
    data are deleted
  IF database or disk access error THEN
    CALL xxx_LogError to write to error log
    CALL drm_PutEvent to post event to database
    SET error drm_status
  ENDIF
  ENDIF
  ENDDO
ENDIF
RETURN
```

### drm\_init—DRM-Initialization Driver Module

#### Parameters:

drm\_config\_info : data\_out

drm\_status : control\_out

**Body:** drm\_init initiates a connection to the LPGS database, retrieves DRM-related configuration information from the database, sets up system signals and signal handler.

## REVIEW

### PDL:

```
CALL xxx_ConnectToDB to access the LPGS database
CALL xdb_get_task_config to retrieve LPGS configuration parameters that are needed
  by DRM
IF database error THEN
  CALL xxx_LogError to send warning messages
  CALL drm_PutEvent to post event to database
  CALL xxx_DisconnectFromDB to disassociate from the LPGS database
  RETURN with error drm_status
ENDIF
CALL xxx_Signal to trap UNIX system signals.
SPECIFY signal handler using xxx_Handler
IF error in setting up signals THEN
  CALL xxx_LogError to send warning messages
  CALL drm_PutEvent to post event to database
  RETURN with error drm_status
ENDIF
RETURN
```

### drm\_main—DRM Driver Module

**Parameters:** None

**Body:** drm\_main is the main module for the DMS Resource Manager process. This module calls the drm\_init module to initialize the process. It then sets a timer for periodic disk clean up. drm\_main then waits for the timer to expire or for reception of messages from operator. When the timer expires, drm\_main calls drm\_clean\_up\_disk and then drm\_check\_disk\_usage. When drm\_main receives an operator request, it calls drm\_proc\_user\_input to process the request. Before drm\_main terminates, it calls xxx\_DisconnectFromDB() to disassociate it from the LPGS database.

### PDL:

```
CALL drm_init to initialize DRM task, and to set up signal handler
IF database error THEN
  CALL xxx_DisconnectFromDB to disassociate from the LPGS database
  TERMINATE drm_main
ENDIF
SET disk-management timer
DOWHILE drm_main is not set to abort or no system break-up signal is captured
  IF user requests for disk management THEN
    CALL drm_proc_user_input to process the requests
    IF database or disk access error THEN
      SET drm_main to abort
    ENDIF
  ENDIF
ENDIF
IF timer expired THEN
```

## REVIEW

```
CALL drm_clean_up_disk to scan the database for mark-up directory and clean
    all data within the directory
IF database or disk access error THEN
    SET drm_main to abort
ENDIF
CALL drm_check_disk_usage to monitor disk usage status
RESET timer for next disk management session
IF disk checking or timer setting error THEN
    SET drm_main to abort
ENDIF
ENDIF
ENDDO
CALL xxx_DisconnectFromDB to disassociate from the LPGS database
```

### **drm\_proc\_user\_input—Process-User-Input Driver Module**

#### **Parameters:**

drm\_config\_info : data\_in

drm\_status : control\_out

**Body:** drm\_proc\_user\_input calls a LPGS library module (xxx\_get\_user\_input) to get operator requests to delete data. It then queries the database to find the associated product request and directories for this deletion. After successfully deleting the data, it updates the LPGS database to reflect this change. Finally, it calls system library module (xxx\_display\_output) to send operator a response with respect to his/her request.

#### **PDL:**

```
CALL xxx_get_user_input to retrieve user's request from GUI
VALIDATE the request
IF user requests to delete a data THEN
    IF the directory of requested data is given THEN
        CALL drm_recursive_deletion to delete data as requested
    ELSE IF the directory of the requested data is not given THEN
        CALL ddb_get_pr_to_del to access the "product request" base table to get the
            directory of the requested data
        CALL drm_recursive_deletion to delete data as requested
    ENDIF
    CALL xdb_update_pr to update product request base table to reflect this change
ENDIF
IF user request to delete a work order or a product request THEN
    CALL ddb_get_pr_to_del to access the database and get all directories associated
        with the requested work order or product request, with the aid from system
        configuration information, drm_config_info
    CALL drm_recursive_deletion to delete all files in the directories and all their sub-
        directories, plus the associated L1 product in the product delivery directory
```

## REVIEW

```
CALL xdb_update_pr to update the product request base table to reflect this change
IF error encountered in any of the above statement THEN
    CALL xxx_LogError to write to error log
    CALL xxx_PutEvent to post event to database
    SET error drm_status
ENDIF
ENDIF
CALL xxx_display_output to notify the user of the status via GUI
RETURN
```

### **drm\_recursive\_deletion—Clean Directory And All Sub-Directories**

#### **Parameters:**

ddb\_del\_pr\_dir : data\_in

drm\_status : control\_out

**Body:** This module deletes all files in drm\_del\_pr\_dir plus all files in its sub-directories. It also cleans up the product delivery directory where the marked L1 product resides. It must verify the completeness of file deletions afterward.

#### **PDL:**

```
VERIFY that all ddb_del_pr_dir exist
VERIFY that L1 product associated with the “product request” in ddb_del_pr_dir exists
    in the product delivery directory
IF any of the file doesn’t exist THEN
    CALL xxx_LogError to write to error log
    CALL drm_PutEvent to post event to database
    RETURN with error drm_status
ENDIF
DELETE files in ddb_del_pr_dir and its sub-directories
DELETE associated L1 product in the product delivery directory
IF deletion error THEN
    CALL xxx_LogError to write to error log
    CALL drm_PutEvent to post event to database
    SET error drm_status
ENDIF
RETURN
```

### **6.3.7 DMS Generate Reports (DGR) Task**

This subsection describes the DGR task software.

#### **6.3.7.1 Task Overview**

The DGR task provides the necessary processing for the LPGS/IAS interface and generates the accounting report for the LPGS system. The DGR task is started by the PSI task and runs as a

## REVIEW

background task. IAS periodically retrieves the characterization statistics from the LPGS database and marks the statistics for deletion. The DGR task deletes the characterization statistics which have been marked for deletion by IAS. Also, the DGR task can receive a request from the operator to delete the characterization statistics. The DGR task periodically stores the LPGS accounting information in the database to create a historical record of the LPGS processing. The DGR task also processes operator requests to generate an accounting report based on the historical information stored in the database.

### 6.3.7.2 Initialization

The DGR task is started by the LPGS Initialization Task. The DGR task initialization includes connecting to the database, creating a signal handler, and retrieving configuration information from the database. Any errors encountered during initialization results in the DGR task exiting.

### 6.3.7.3 Normal Operation

The DGR task runs when the one of the two task timers expires or when an operator request is received. When the IAS interface timer expires, the DGR task deletes the characterization trending statistics which were marked for deletion by IAS. When the other timer expires, this task stores the historical accounting information in the database. When an operator request is received, this task either provides the operator with the requested accounting report or deletes the characterization statistics.

### 6.3.7.4 Error Handling

The DGR task establishes a signal handler to capture all fatal UNIX signals which would cause the task to abort. The signal handler sends an error message to the operator, disconnects from the database, and gracefully shuts down the task. Nonfatal errors are logged to the LPGS error log and an error notification is sent to the user interface.

### 6.3.7.5 Design

This subsection presents the design of the DGR task. The structure chart for the task is illustrated in Figure 6-9. The module specifications for the DGR task are provided (in alphabetical order) below.

#### **dgr\_calc\_timer—Calculate Wakeup Time**

##### **Parameters:**

dgr\_timer\_value : data\_out

dgr\_status : control\_out

**Body:** The DGR process needs to periodically provide the processing necessary to support the IAS interface. Also, the DGR process periodically collects the current LPGS statistics. The dgr\_calc\_timer module determines which event will occur next. Then, this module calculates when the DGR process will need to wakeup to process the next event.

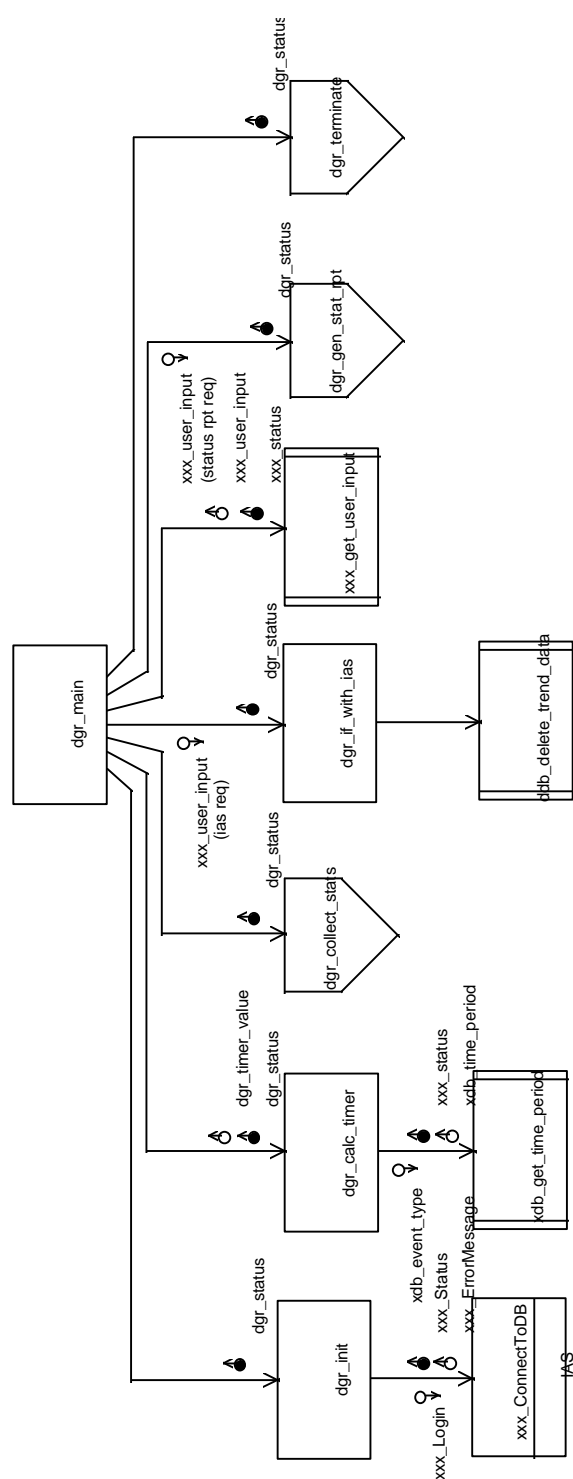


Figure 6-9. DGR Structure Chart (1 of 4)

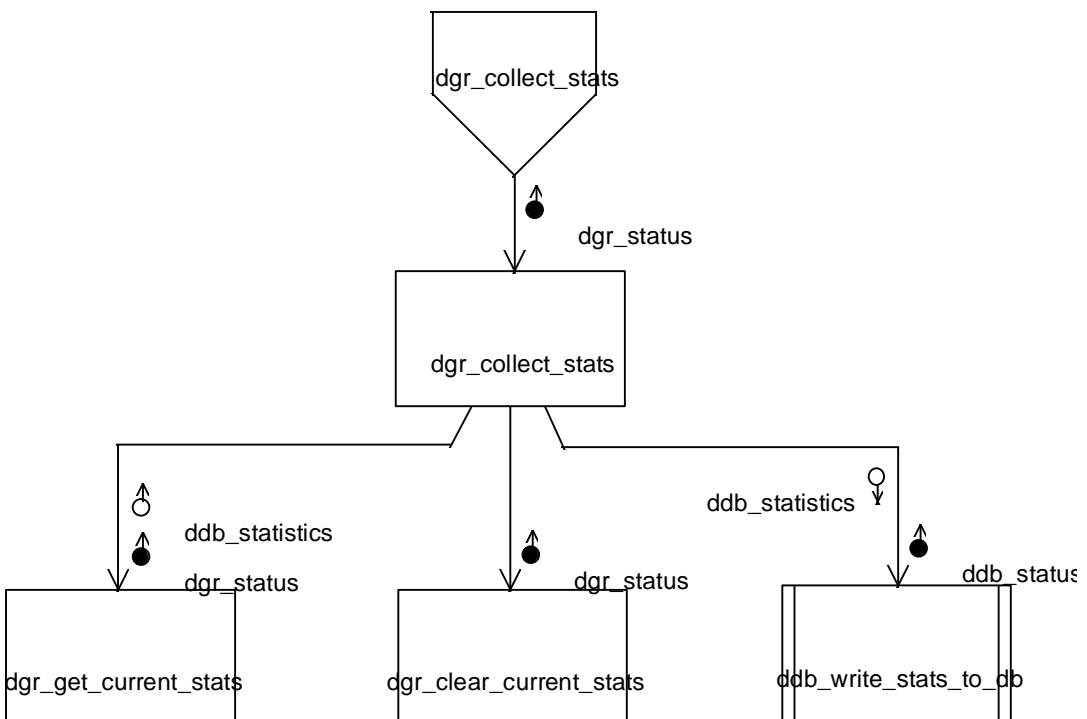


Figure 6-9. DGR Structure Chart (2 of 4)

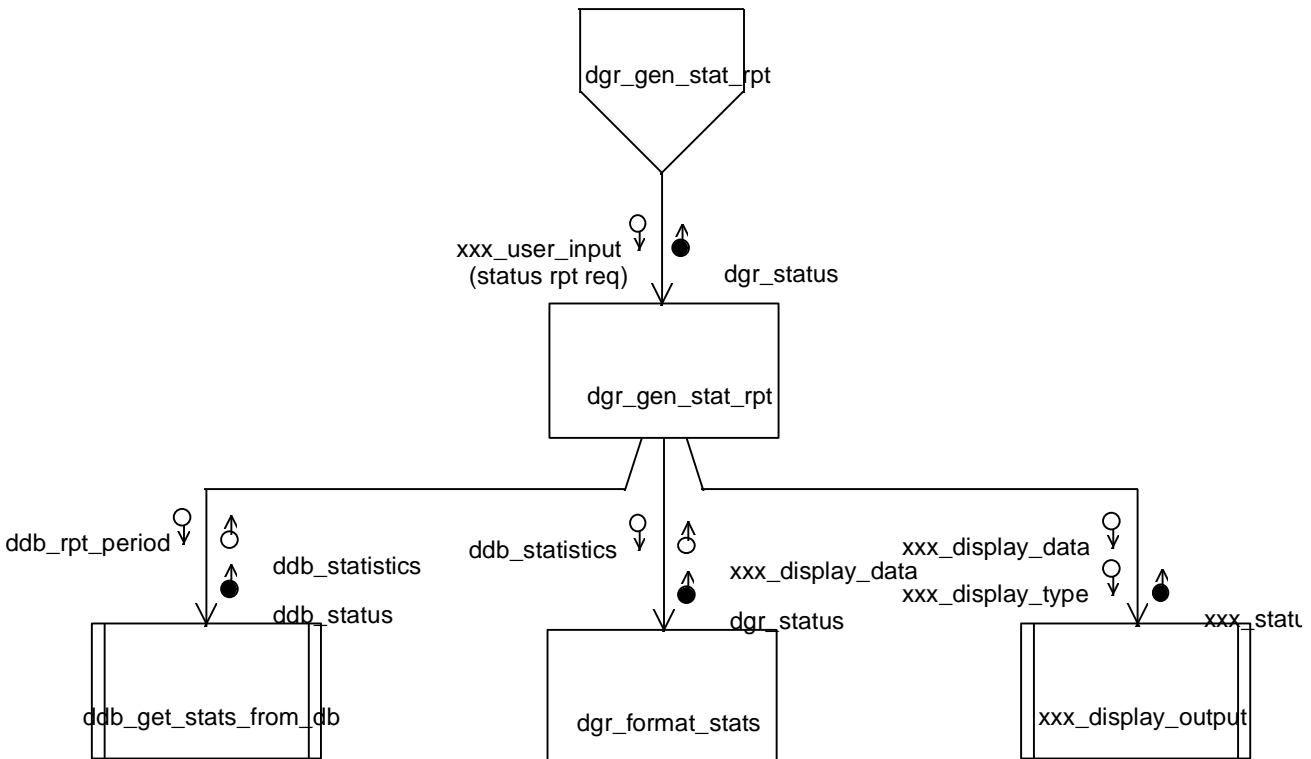


Figure 6-9. DGR Structure Chart (3 of 4)

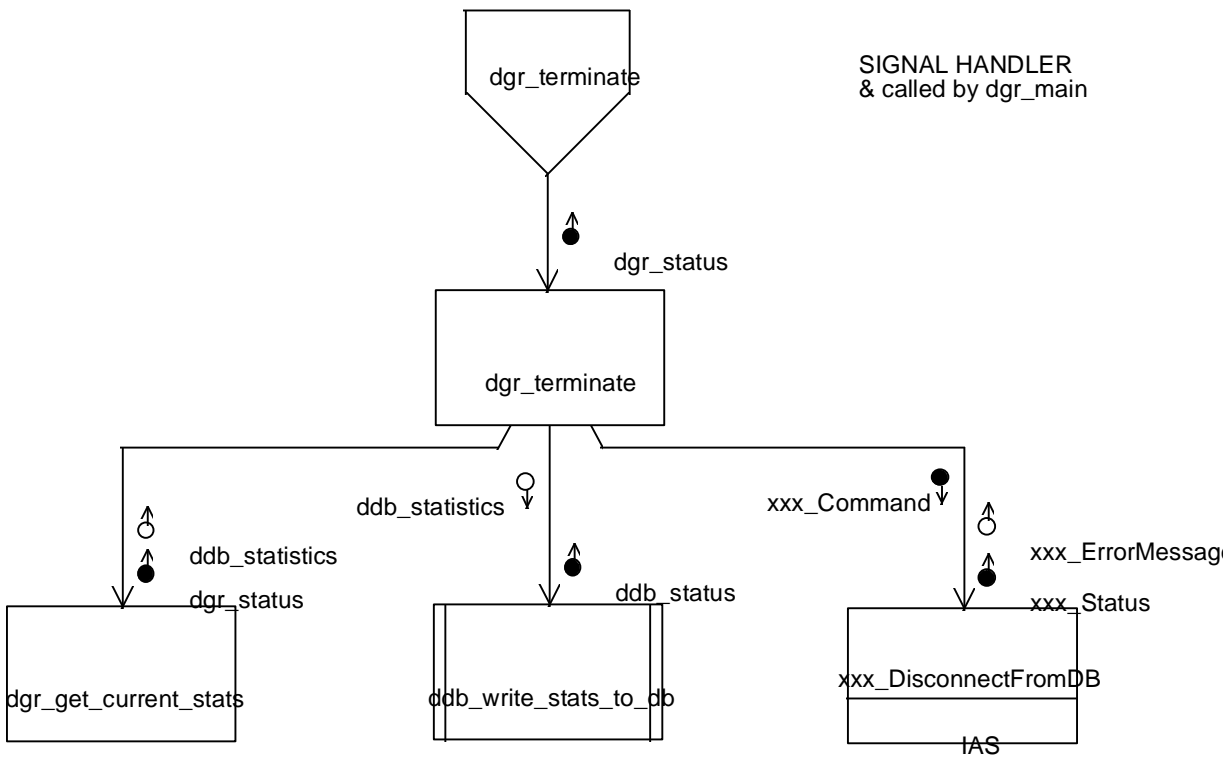


Figure 6-9. DGR Structure Chart (4 of 4)

## REVIEW

### **dgr\_clear\_current\_stats—Clear The Current Statistics In Global Memory**

#### **Parameters:**

dgr\_status : control\_out

**Body:** This module clears all of the counts in global memory. Then, this module unlocks the global memory.

### **dgr\_collect\_stats—Collect Statistics**

#### **Parameters:**

dgr\_status : control\_out

**Body:** All of the subsystems within LPGS will update a count within global memory when an event associated with the count occurs. (re: ingested 3 scenes) This module is called periodically to store the counts in global memory into the database, thereby creating a historical record of the LPGS system. When this module is called, it calls a module to get the current statistics from global memory. Then, this module calls a module to clear the current statistics. Finally, this module calls a database access module to store the statistics in the database.

### **dgr\_format\_stats—Format The Statistics Into A Report**

#### **Parameters:**

xxx\_display\_data : data\_out

ddb\_statistics : data\_in

dgr\_status : control\_out

**Body:** This module receives as input the raw statistics for the reporting period that were stored in the database. This module converts the statistics into the format necessary for the report. This module returns the statistic report.

### **dgr\_gen\_stat\_rpt—DGR Generate Statistic Report**

#### **Parameters:**

dgr\_status : control\_out

xxx\_user\_input : data\_in

**Body:** This module is invoked when the operator requests a statistic report. The xxx\_user\_input specifies the reporting period for the statistics the operator wants to view. This module calls a database access module to get the statistic. Then, this module calls a module to format the report. Finally, this module calls a global library routine to display the report for the operator.

## REVIEW

### **dgr\_get\_current\_stats—Get The Current Statistics From Global Memory**

#### **Parameters:**

ddb\_statistics : data\_out

dgr\_status : control\_out

**Body:** This module locks global memory to prevent any updates from occurring. Then, this module gets the current statistics from global memory and returns the statistics.

### **dgr\_if\_with\_ias—Interface with IAS**

#### **Parameters:**

dgr\_status : control\_out

xxx\_user\_input : data\_in

**Body:** This module provides the necessary processing for the IAS interface. IAS will access the LPGS database to obtain the trending data statistics. IAS will retrieve the statistics on a periodic basis. The LPGS processing necessary for this interface is limited to deleting the old trending data from the database. This module calls a database access routine to delete the old trending data from the database.

### **dgr\_init—DGR Initialization function**

#### **Parameters:**

dgr\_status : control\_out

**Body:** The dgr\_init module connects to the database, sets up a signal handler, and allocates the global memory for the statistics. This module returns a bad status if an error occurs during processing.

### **dgr\_main—DMS Generate Report Main Module**

#### **Parameters:** None

**Body:** dgr\_main is the main module for the DMS Generate Report process. This module calls the dgr\_init to initialize the process. Then, this module calls a module to determine the next time the process will need to wake up. Then, dgr\_main waits for the timer to expire or for the receipt of a message from the operator. When the timer expires, this module either calls a module to collect the statistics for the current reporting period or a module which provides the processing necessary for the IAS interface. When an operator request is received, dgr\_main calls a module to generate the statistics report or the module that provides the IAS interface.

#### **PDL:**

```
CALL dgr_init to initialize the process
DOWHILE dgr_status is good
    CALL dgr_calc_timer to determine when to wake up
```

## REVIEW

```
IF dgr_status is good THEN
    Set the timer to expire in dgr_timer_value time
    Wait for the timer to expire or for input from the user
    IF the timer expired THEN
        IF it is time to collect the statistics THEN
            CALL dgr_collect_stats to collect the LPGS statistics
        ELSE
            IF it is time to automatically provide the IAS interface THEN
                CALL dgr_if_with_ias to provide processing for the IAS interface
            ENDIF
        ENDIF
    ENDIF
ENDIF
IF process was notified of user input THEN
    CALL xxx_get_user_input to get the user's request
    IF xxx_status returned from xxx_get_user_input is good THEN
        IF xxx_user_input was a IAS interface request THEN
            CALL dgr_if_with_ias to process the IAS request
        ELSE
            IF xxx_user_input was a statistics report request THEN
                CALL dgr_gen_stat_rpt to generate the statistic report
            ENDIF
        ENDIF
    ELSE
        IF xxx_status error indicates the task should terminate THEN
            Set dgr_status to indicate the error
        ENDIF
    ENDIF
ENDIF
ENDIF
ENDDO
CALL dgr_terminate to terminate the dgr_main process
EXIT
```

### **dgr\_terminate—Terminate the DGR Process**

#### **Parameters:**

dgr\_status : control\_out

**Body:** This module terminates the DGR process. This module stores the statistics in global memory in the database. Then, this module disconnects from the database and de-allocates the LPGS global memory. Finally, this module terminates the DGR process.

### **6.3.8 Subsystem Utilities**

This subsection presents the design of the DMS utilities. The module specifications for the utilities follow.

## REVIEW

### **ddb\_catalog\_l0r—Catalog L0R Products**

**Parameters:**

xxx\_prod\_req\_id : data\_in

xxx\_prod\_req\_dir : data\_in

ddb\_status : control\_out

**Body:** This module catalogs the L0R product files into the database.

### **ddb\_delete\_trend\_data—Delete Trending Data**

**Parameters:** None

**Body:** This database access module deletes the trending data statistics from LPGS. The operator can specify how old the trending data must be before it is deleted. This module, which is called periodically, deletes the old trending data.

### **ddb\_find\_prod\_req—Find The Product Request**

**Parameters:**

ddb\_msg\_pr\_id : data\_in

ddb\_status : control\_out

xxx\_prod\_req\_id : data\_out

**Body:** This module searches the database for a product request with the specified product request id.

### **ddb\_get\_next\_prod—Get Next Product Request**

**Parameters:**

ddb\_status : control\_out

xxx\_prod\_req : data\_out

xxx\_prod\_req\_id : data\_out

**Body:** This module queries the database to determine the next product request which needs the L0R product ingested. Then, this module calls xdb\_get\_prod\_req to read the product request from the database.

### **ddb\_get\_next\_xmit—Get Next Product Request To Be Transmitted**

**Parameters:**

ddb\_status : control\_out

xxx\_prod\_req\_id : data\_out

## REVIEW

**Body:** This module queries the database to determine the next product request, which needs the L1 product transmitted to ECS.

### **ddb\_get\_pr\_to\_del—Get Product Requests To Delete**

#### **Parameters:**

drm\_config\_info : data\_in

ddb\_del\_pr\_dir : data\_out

ddb\_stat : control\_out

**Body:** This module accesses the database, retrieves a list of L1 product requests which are marked to be deleted. It then returns a list of data directories associated with the marked product requests to drm\_clean\_up\_disk. In case of product-request data deletion issued by operator, this module queries the database and returns the name of associated directories to drm\_proc\_user\_input.

#### **PDL:**

```
DOFOR all product requests in base table "product_request"
  SEARCH for "eligible for deletion" mark
  IF (database error) THEN
    CALL xxx_LogError() to send warning message
    SET error ddb_status
    EXIT loop
  ELSE
    IF (the product request is marked for deletion) THEN
      DOFOR all directories associated with the product request
        IF (directory exists) THEN
          STORE the marked directory in an array, ddb_del_pr_dir
          ARCHIVE (tar) the marked directory to a hold directory (TBD)
          IF (I/O error archiving the directory) THEN
            CALL xxx_LogError() to send warning message
          ENDIF
          CALL xxx_Put_Event() to write to event log
        ELSE
          CALL xxx_LogError() to send warning message
        ENDIF
      ENDDO
    ENDIF
  ENDDO
RETURN
```

## REVIEW

### **ddb\_get\_stats\_from\_db—Get Statistics From The Database**

#### **Parameters:**

ddb\_rpt\_period : data\_in

ddb\_status : control\_out

ddb\_statistics : data\_out

**Body:** This database access module gets the statistics from the database for the specified reporting period. This module gets all of the statistics that were collected during the reporting period. This module returns the statistics in an array.

### **ddb\_insert\_prod\_req—Enter data into database**

#### **Parameters:**

xxx\_prod\_req\_id : data\_in

xxx\_prod\_req : data\_in

ddb\_status : control\_out

**Body:** This module inserts the data passed to it into the product request database table.

### **ddb\_write\_stats\_to\_db—Write Statistic To The Database**

#### **Parameters:**

ddb\_statistics : data\_in

ddb\_status : control\_out

**Body:** This module writes the statistics for the last reporting period to the next entry in the historical statistic database table.

## Section 7. Radiometric Processing Subsystem

### 7.1 Introduction

The RPS processes L0R image data to produce Level 1 radiometrically corrected (L1R) images.

The RPS is being developed external to the LPGS development by the Algorithm Implementation Team (AIT) for IAS and LPGS. RPS is considered a black box by the LPGS implementation team.

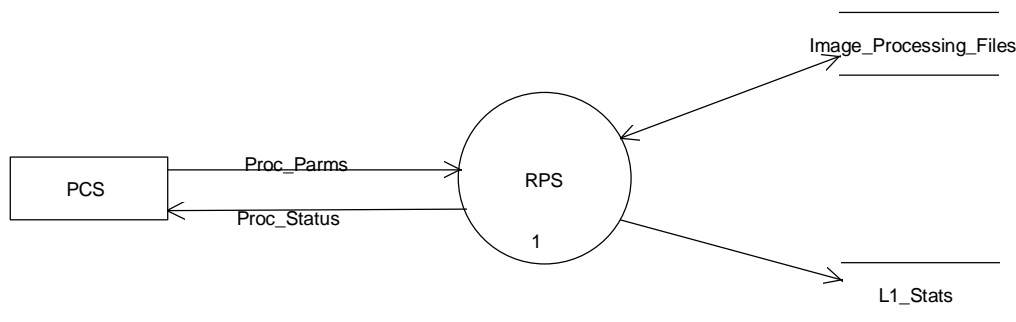
Information presented in this section is intended to provide the LPGS development team's understanding of the RPS. (For further information on the RPS design, refer to the IAS Critical Design Specification, Reference 11.)

### 7.2 Design Overview

This section provides an overview of the RPS software design. The relationship between the RPS and other LPGS subsystems is presented, along with a discussion of the considerations and assumptions used in the design process.

#### 7.2.1 Subsystem Software Overview

Figure 7-1 contains the context diagram of the RPS.



**Figure 7-1. RPS Context Diagram**

The RPS interfaces with the PCS and LPGS system files. The RPS receives processing parameters (Proc\_Parms) from the PCS and returns processing status (Proc\_Status) to PCS. The RPS writes completed L1R image data to the work order directory. The RPS files are accessed by the QAS for quality assessment and AAS for anomaly analysis.

The RPS performs radiometric calibration and L1R processing. The purpose of radiometric calibration is to determine the calibration in-flight of each detector, that is, the conversion from digital number to absolute radiance. The primary purpose of the L1R processing is to convert the

## **REVIEW**

brightness of the image pixels to absolute radiance, and is done prior to Level 1 geometric processing. This process uses various ground and in-flight determined calibrations. Another function of L1R processing is to characterize the quality and various features of the data.

### **7.2.2 Design Considerations**

The RPS is being developed initially for, and under the direction of, the IAS project by the AIT. The LPGS system engineers are monitoring the development of the RPS to ensure that the LPGS's requirements are being met where they differ from the IAS requirements. Differences between the LPGS and IAS requirements for RPS are minor.

### **7.2.3 Design Assumptions**

The following assumptions were made in reusing the RPS:

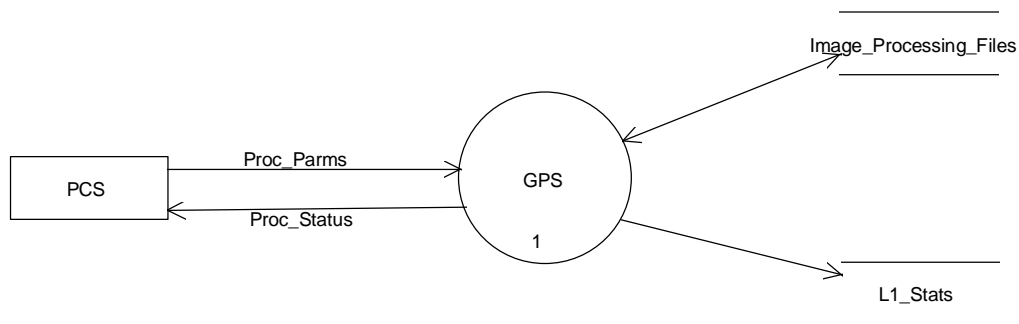
- The RPS will seamlessly integrate into the LPGS.
- The RPS will include all radiometric processing functionality required by the LPGS.
- Each LPGS product request will have a unique directory that contains three or more subdirectories: the input directory, the save directory, and a work order directory for each time the work is run. RPS will be able to access these directories.
- The RPS will be able to meet LPGS processing performance requirements for production of the L1R images.
- The RPS will meet LPGS processing volume requirements for the production of the L1R images.

## Section 8. Geometric Processing Subsystem

---

### 8.1 Introduction

The GPS provides all the functionality required for Level 1G product generation. This system implements all of the LPGS geometry algorithms. Figure 8-1 contains the context diagram of the GPS. For more details on the GPS design, see the IAS Geometric Processing Subsystem Detailed Design Specification (Requirements Document 14).



**Figure 8-1. GPS Context Diagram**

# Section 9. Quality Assessment Subsystem

---

## 9.1 Introduction

The QAS generates and assembles information about image artifacts and effects, and produces a summary of the processed image quality. The QAS performs automated quality assessment after radiometric and geometric correction of the images. The QAS is also responsible for providing the manual visual inspection system for viewing the corrected images during intermediate LPGS processing and for viewing the final product.

## 9.2 Design Overview

This section provides an overview of the QAS software design. It presents the relationships between the QAS and the other LPGS subsystems. It also discusses the assumptions, constraints, and considerations used in the design process.

### 9.2.1 Subsystem Software Overview

Figure 9-1 contains the QAS context diagram. The QAS is composed of two automated, image quality assessment software tasks that are invoked by the PCS task, and a third user interactive task for viewing images and quality reports. The two automated tasks assess the intermediate image following the radiometric and geometric correction process, respectively. When notified that L1 radiometric or geometric processing is complete, the PCS passes processing parameters via an ODL file to the QAS task. The QAS reads the L1R and L1G characterization and correction results, and compares these values to the threshold values identified in the ODL file. If the image results exceed a threshold or a combination of thresholds, the QAS sets the corresponding threshold flag and notifies the PCS. Also, the QAS stores a summary of the quality of the image processing in the LPGS database. The third task which provides for the visual inspection will be implemented using a COTS package.

### 9.2.2 Design Considerations

This section describes the design considerations used while developing the QAS.

#### 9.2.2.1 Design Assumptions

The following assumptions were made during the design of the QAS:

- The QAS allows the operator to modify parameter thresholds prior to the start of L1 image processing.
- COTS products will be used for
  - Viewing and printing the intermediate and final images
  - Viewing and printing the online quality report

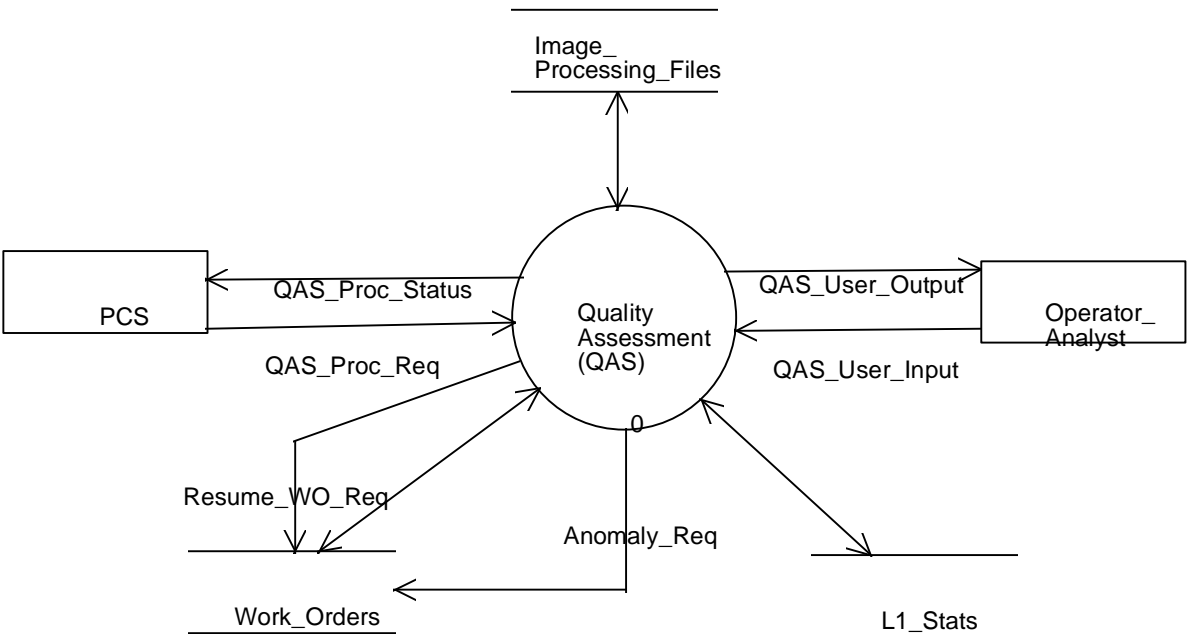


Figure 9-1. QAS Context Diagram

## REVIEW

- Reading files in ODL format
- Accessing the database

### 9.2.2.2 Open Issues

The selection of the COTS package, which is needed for viewing the image in different formats, has not been made.

### 9.2.3 Subsystem Error Handling

The following lists the types of errors that the QAS might encounter and the action required:

Error	Action Taken
Cannot connect to LPGS database	Report error to the PCS via status message, write error to error log, and terminate
Cannot read/update the LPGS database	Report error to the PCS via status message, write error to error log, and terminate
Cannot read ODL file	Report error to the PCS via status message, write error to error log, and terminate

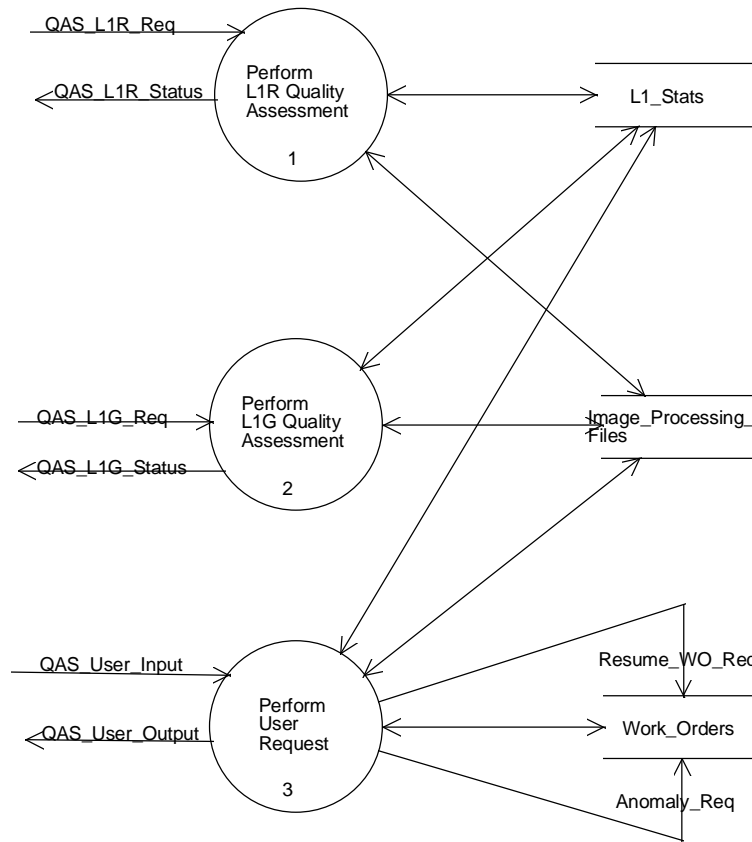
## 9.3 Subsystem Design

The QAS is broken into three tasks: the L1R Quality Assessment (Q1R) for assessing the quality of the radiometric corrected image, the L1G Quality Assessment (Q1G) for assessing the quality of the geometrically corrected image, and the Quality Assessment User Interface (QUI). The QUI is used to visually inspect the LPGS intermediate image and the final product. The Q1R and Q1G run in script mode, and QUI is interactive. The QAS data flow diagram is shown in Figure 9-2.

Potential radiometry image artifacts to compare to a nominal or expected value are

- Dropped lines
- Impulse noise
- Memory effect
- Scan correlated shift
- Detector saturation
- Coherent noise
- Random noise
- Histogram analysis
- Detector inoperability

## REVIEW



**Figure 9-2. QAS Level 0 DFD**

Potential geometric image artifacts to assess and validate a nominal or expected value are:

- Ephemeris data
- Spacecraft attitude data
- Gyro data
- Gyro drift data
- ADS data
- Forward Scan Error
- Reverse Scan Error
- Image corners and center point
- Scan start time
- Scan direction

## REVIEW

- Line length count
- Scan gap

A COTS package will be used to view and print the quality report; and to view and print the intermediate and final images.

### 9.3.1 L1R Quality Assessment Task

#### 9.3.1.1 Task Overview

The Level 1 Radiometric Quality Assessment task (Q1R) analyzes the results from the Level 1 radiometric correction and characterization process. The output from the radiometric algorithms are the data input for the automated Q1R analysis. A combination of characterization and correction thresholds, and results are the basis for the quality assessment applied to the radiometrically corrected image.

#### 9.3.1.2 Initialization

The Q1R task is started by the PWC task. It connects to the database and opens the image file to retrieve radiometric correction and characterization results.

#### 9.3.1.3 Normal Operation

The PWC task invokes the Q1R task. The PWC passes processing parameters and pointers associated with the WO to start Q1R processing. Following a successful initialization, the Q1R imports data from the L1R image file and the LPGS database. The output and basis for the Q1R analysis are from the Level 1 radiometric algorithms used in the radiometric characterization and correction process. A combination or singular L1R threshold and/or characterization will determine the image's quality. If it is determined that the image has no anomalies, a successful Q1R assessment status is returned and image processing proceeds. If the image contains radiometric anomalies, the AAS is alerted via the database and the image is passed to the AAS for further analysis. All interfaces to external subsystems are via the PCS and the database. The results of the Q1R analysis are stored in the QAS\_Results database table.

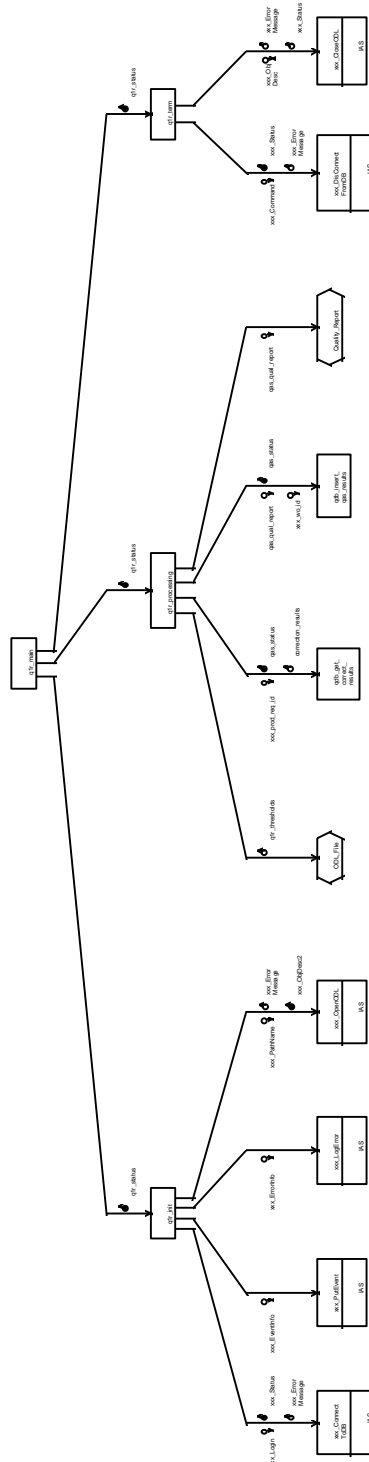
#### 9.3.1.4 Design

This subsection presents the design of the Q1R task. The structure chart for the task is illustrated in Figure 9-3. The module specifications for the Q1R task are provided below (alphabetically):

##### **q1r\_init—Level 1 Radiometric Initialization**

**Body:** This module opens the ODL file, connects to the database, and initializes the Level 1 Radiometric data fields. If a problem is encountered it writes the error to the error log and sets the fail return code. If the connection is successful, the success return code is set.

## REVIEW



**Figure 9-3. Q1R Structure Chart**

## REVIEW

### PDL:

```
CALL xxx_OpenODL to open ODL file
IF fail THEN
    CALL xxx_LogError
    SET q1r_status to fail
    RETURN
ENDIF
CALL xxx_ConnectToDB to connect to database
IF success THEN
    INITIALIZE Level 1 Radiometric quality assessment fields
    SET q1r_status to success
ELSE
    CALL xxx_LogError
    SET q1r_status to fail
ENDIF
RETURN
```

### q1r\_main—Level 1 Radiometric Main

**Body:** This module is the Level 1 Radiometric function initiator. The module begins by checking for the presence of the Object Description Language (ODL) file. If no ODL file is detected then processing is halted and the fail return code is set. Otherwise, this module sequentially invokes the Level 1 Radiometric assessment functions.

### PDL:

```
CALL q1r_init to open the ODL file, initialize variables and connect to database
IF return code from q1r_init was success THEN
    CALL q1r_processing to compare values
ENDIF
CALL q1r_term to close database connection and ODL file
EXIT
```

### q1r\_processing—Level 1 Radiometric Processing

**Body:** This module accesses the L1\_Stats database tables to read the results from the Level 1 Radiometric Processing. These results are compared to Radiometric correction thresholds in the ODL File. Differences or deltas between the results and the thresholds are calculated. A single, or combination of deltas, and boundary conditions determine the quality of the L1R image. If the results of the comparison determine the image contains anomalies, then a status of fail is returned. Otherwise, a status of success is returned. Results of the assessment (qas\_qual\_report) are written to the Quality\_Report and the database.

### PDL:

```
DOFOR each Level 1 Radiometric Processing correction results
    CALL qdb_get_correct_results to read qas_correction_results from the L1_Stats
    database tables
```

## REVIEW

```
IF not success THEN
    Set q1r_status to fail
    RETURN
ENDIF
READ q1r_thresholds from the ODL file
IF not success THEN
    SET q1r_status to fail
    RETURN
ENDIF
CALCULATE delta
IF delta is not within bounds THEN
    STORE delta
ENDIF
ENDDO
ASSESS image quality
IF image is "poor" quality THEN
    SET q1r_status to fail
ELSE
    SET q1r_status to success
ENDIF
CALL qdb_insert_gas_results to write gas_qual_report to the database
WRITE gas_qual_report to Quality_Report
RETURN
```

### **q1r\_term—Level 1 Radiometric Termination**

**Body:** This module closes the ODL file and disconnects from the database.

**PDL:**

```
CALL xxx_CloseODL to close the ODL file
CALL xxx_DisConnectFromDB to disconnect from the database
RETURN
```

## **9.3.2 L1G Quality Assessment Task**

### **9.3.2.1 Task Overview**

The Level 1 Geometric Quality Assessment task (Q1G) analyzes the results from the Level 1 geometric correction. The output from the geometric algorithms are the data input to the automated Q1G analysis. A combination of correction thresholds will serve as the basis for the quality assessment applied to the geometrically corrected image.

### **9.3.2.2 Initialization**

The Q1G task is started by the PWC task. The Q1G connects to the database and opens the image file to retrieve the geometric correction results.

## REVIEW

### 9.3.2.3 Normal Operation

The PWC task invokes the Q1G task. The PWC task passes processing parameters and pointers associated with the WO to start Q1G processing. Following a successful initialization, the Q1G imports data from the L1G image file and the LPGS database. The output and basis for the Q1G analysis are from the Level 1 geometric algorithms used in the geometric correction process. A combination or singular L1G threshold and/or characterization determines the image's quality. If it is determined that the image has no anomalies a successful Q1G status is returned and image processing proceeds. If the image contains geometric anomalies, the AAS is alerted via the database and the image is passed to the AAS for further analysis. All interfaces to external subsystems is via the PCS and the database. All results of the Q1G analysis are stored in the QAS\_Results database table.

### 9.3.2.4 Design

This subsection presents the design of the Q1G task. The structure chart for the task is illustrated in Figure 9-4. The module specifications for the Q1G task are provided below (alphabetically):

#### **q1g\_init—Level 1 Geometric Initialization**

**Body:** This module opens the ODL file, connects to the database, and initializes the Level 1 Geometric data fields. If a problem is encountered it writes the error to the error log and sets the fail return code. If the connection is successful, the success return code is set.

**PDL:**

```
CALL xxx_OpenODL to open ODL file
IF fail THEN
    CALL xxx_LogError
    SET q1g_status to fail
    RETURN
ENDIF
CALL xxx_ConnectToDB to connect to database
IF success THEN
    INITIALIZE Level 1 Geometric quality assessment fields
    SET q1g_status to success
ELSE
    CALL xxx_LogError
    SET q1g_status to fail
ENDIF
RETURN
```

#### **q1g\_main—Level 1 Geometric Main**

**Body:** This module is the Level 1 Geometric function initiator. The module begins by checking for the presence of the Object Description Language (ODL) file. If no ODL file is detected then processing is halted and the fail return code is set. Otherwise, this module sequentially invokes the Level 1 Geometric assessment functions.

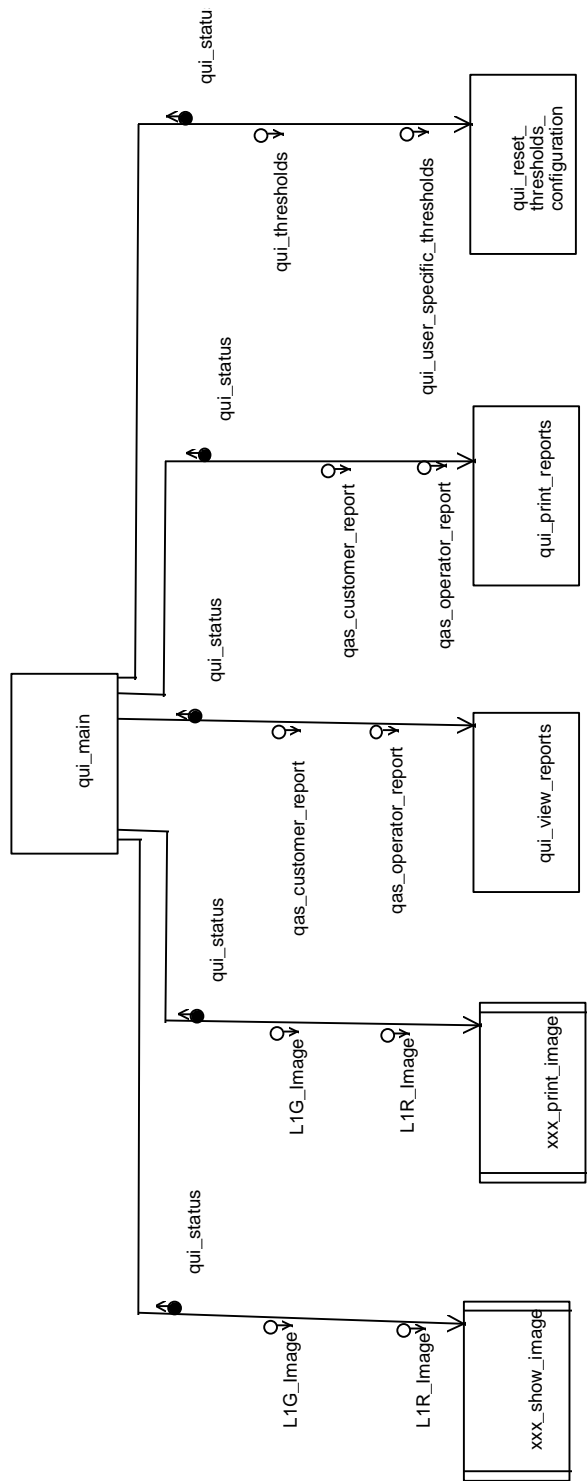


Figure 9-4. Q1G Structure Chart

## REVIEW

### PDL:

```
CALL q1g_init to open the ODL file, initialize variables and connect to database
IF return code from q1g_init was success THEN
    CALL q1g_processing to compare values
ENDIF
CALL q1g_term to close database connection and ODL file
EXIT
```

### q1g\_processing—Level 1 Geometric Processing

**Body:** This module accesses the L1\_Stats database tables to read the results from the Level 1 Geometric Processing. These results are compared to geometric correction thresholds in the ODL File. Differences or deltas between the results and the thresholds are calculated. A single, or combination of deltas, and boundary conditions determine the quality of the L1G image. If the results of the comparison determine the image contains anomalies, then a status of fail is returned. Otherwise, a status of success is returned. Results of the assessment (qas\_qual\_report) are written to the Quality\_Report and the database.

### PDL:

```
DOFOR each Level 1 Geometric Processing correction results
    CALL qdb_get_correct_results to read correction_results from the L1_Stats
        database tables
    IF not success THEN
        Set q1g_status to fail
        RETURN
    ENDIF
    READ q1g_thresholds from the ODL file
    IF not success THEN
        SET q1g_status to fail
        RETURN
    ENDIF
    CALCULATE delta
    IF delta is not within bounds THEN
        STORE delta
    ENDIF
ENDDO
ASSESS image quality
IF image is "poor" quality THEN
    SET q1g_status to fail
ELSE
    SET q1g_status to success
ENDIF
CALL qdb_insert_qas_results to write qas_qual_report to the database
WRITE qas_qual_report to the Quality_Report
RETURN
```

## REVIEW

### **q1g\_term—Level 1 Geometric Termination**

**Body:** This module closes the ODL file and disconnects from the database.

**PDL:**

```
CALL xxx_CloseODL to close the ODL file
CALL xxx_DisConnectFromDB to disconnect from the database
RETURN
```

### **9.3.3 Quality Assessment User Interface Task**

#### **9.3.3.1 Task Overview**

The QUI task provides the mechanism to perform visual inspections of the intermediate and final images. A COTS package will be used for viewing the images. Oracle Forms will be used to provide the user interface.

#### **9.3.3.2 Initialization**

Initialization will be per the vendor's recommended instructions.

#### **9.3.3.3 Normal Operation**

The QUI task consists of the software necessary for the COTS product to be integrated into the LPGS system. The functionality required to view the images will be through the COTS.

#### **9.3.3.4 Design Input, Output, Algorithm**

The input, output, and algorithm used will be dependent on the COTS package chosen for viewing the images.

# Section 10. Anomaly Analysis Subsystem

---

## 10.1 Introduction

The purpose of the AAS is to provide the tools to investigate problems encountered during LPGS production processing or problems reported by the end user after receipt of the L1 product. This section provides an overview of the AAS software design. The relationship between the AAS and other LPGS subsystems is presented, along with a discussion of the assumptions, constraints, and considerations used in the design process.

## 10.2 Design Overview

### 10.2.1 Design Methodology

The design of the AAS is depicted with DFDs instead of structure charts because most of the AAS capabilities are initiated by the AAS analyst through the UI. The AAS functions and data flows are presented in the AAS DFDs. Many of these functions will be implemented through COTS products.

The AAS consists of a collection of tools that the analyst accesses through a series of user interface screens. Additional information about the AAS UI can be found in Section 10 and in the LPGS User's Guide.

### 10.2.2 Solution Strategy

The AAS software consists of an integrated set of COTS software packages combined with supplementary application software. A TBD COTS product will be used to support the viewing capability for displaying remote sensing images. Oracle Forms will provide the user interface and provide access to the LPGS database.

### 10.2.3 Design Considerations

This section describes the design considerations used while developing the detailed design for the AAS.

#### 10.2.3.1 Assumptions

The following assumptions were made when designing the AAS:

- A. Access to ECS Trouble Ticket System—The AAS analyst will have the ability to access the ECS trouble ticket system using REMEDY.
- B. Notification of Trouble Tickets—The AAS analyst will be notified of ECS trouble tickets assigned to LPGS via e-mail or by phone. The analyst will be provided a trouble ticket number that can be used to retrieve the trouble ticket information using REMEDY.

## REVIEW

- C. Scope of Responsibility—The AAS will resolve production-related problems only. Nonproduction problems will be filtered out and entered into the ECS trouble ticket system for LPGS-detected problems (as a trouble ticket) or for customer-detected problems (as a trouble ticket response).

### 10.2.3.2 Open Issues

There are several open issues that may influence the design of the AAS. The following open issues have been identified:

- A. Selection of the COTS product for viewing images is TBD—The final selection of the COTS product for viewing images has not been made because there is no product that currently supports the writing of the FAST-C format.
- B. Acquisition of returned products—In some cases, the end users may be returning products associated with trouble tickets. The mechanism for transferring the returned products to the LPGS has not been finalized. It is assumed that the ECS will be responsible for transferring the data from media (tape or CD) to disk.
- C. ECS User Services resolution of problems relating to errors in specifying user options—Most of the demand for Landsat data reprocessing at EOSAT is the result of errors in specifying user options. In some cases, the error is the result of data entry mistakes made by User Services personnel. In other cases, the specification error is made by the user. The open issue relating to the AAS is whether user option specification errors that result in reprocessing are resolved in ECS User Services or forwarded to the AAS in the form of trouble tickets. The LPGS systems engineers expect that user option specification errors will be resolved by ECS User Services. Thus, errors of this type will result in ECS User Services issuing a new L1 product generation request with the correct specification of user options. This type of reprocessing would be transparent to LPGS (i.e., it would look to LPGS like any other request) and would not impact the AAS operations in any way. Wider discussion of this issue is needed to confirm that LPGS expectations are correct.

### 10.2.4 Software Reuse Strategy

The AAS is designed to minimize investments in custom software development. The primary functionality needed for the AAS will be provided through COTS software products, which can be customized to support the AAS specific applications.

The AAS analyst uses the results of radiometric and geometric processing performed by RPS and GPS. RPS and GPS contain alternate production procedures that allow the AAS analyst to set more flags to stop processing at discrete points to permit viewing of the results. The image processing support needed by the AAS analyst to diagnose anomalies is provided by RPS and GPS. As a result of this reuse of RPS and GPS, no additional image processing routines other than those provided by the COTS image viewing product are needed to support the AAS.

## REVIEW

### 10.3 Subsystem Design

#### 10.3.1 Functional Overview

This section describes the various functions of the AAS. Figure 10-1 presents the AAS context diagram, which provides a high-level view of the AAS. The AAS analyst represents the primary interface because the AAS is a collection of interactive tools. The AAS also interfaces with the ECS Trouble Ticket System for sharing information about reported problems. The AAS interfaces to the other LPGS subsystems through the database.

Figure 10-2 presents the AAS Level 0 DFD, which identifies the high-level AAS functions. These functions include

- Track Anomalies
- Perform Anomaly Runs
- View LPGS Data
- Generate History Report
- Transmit Anomalies

#### 10.3.2 Track Anomalies

The AAS uses the Anomalies database table to track the status of problem investigations. This table supports tracking of problems detected by LPGS during processing and problems reported by the customer through trouble tickets. The Anomalies table serves as a work-off list for the analyst. Figure 10-3 shows the Level 1 DFD, which identifies the subfunctions provided for tracking problems.

##### 10.3.2.1 Receive LPGS Anomaly

The Receive LPGS Anomaly function allows the analyst to add new LPGS production problems to the Anomalies table. The PWC and QUI tasks are responsible for identifying problems detected during script processing or during visual quality assessment. They notify the analyst by setting the work order state to indicate that it needs the AAS analyst's attention. The AAS analyst selects one of these work orders, and this function fills in the applicable fields on a form. The analyst can make any additional modifications before the new problem is inserted into the Anomalies table. Once the new entry has been added, this function updates the Work\_Orders table to indicate that the problem has been recorded and the AAS is currently investigating it.

##### 10.3.2.2 Receive Trouble Ticket

The Receive Trouble Ticket function allows the AAS analyst to retrieve the trouble ticket from the ECS Trouble Ticket System using REMEDY. The analyst queries the ECS Trouble Ticket System for the desired trouble ticket and manually enters the information into the Anomalies table through a form.

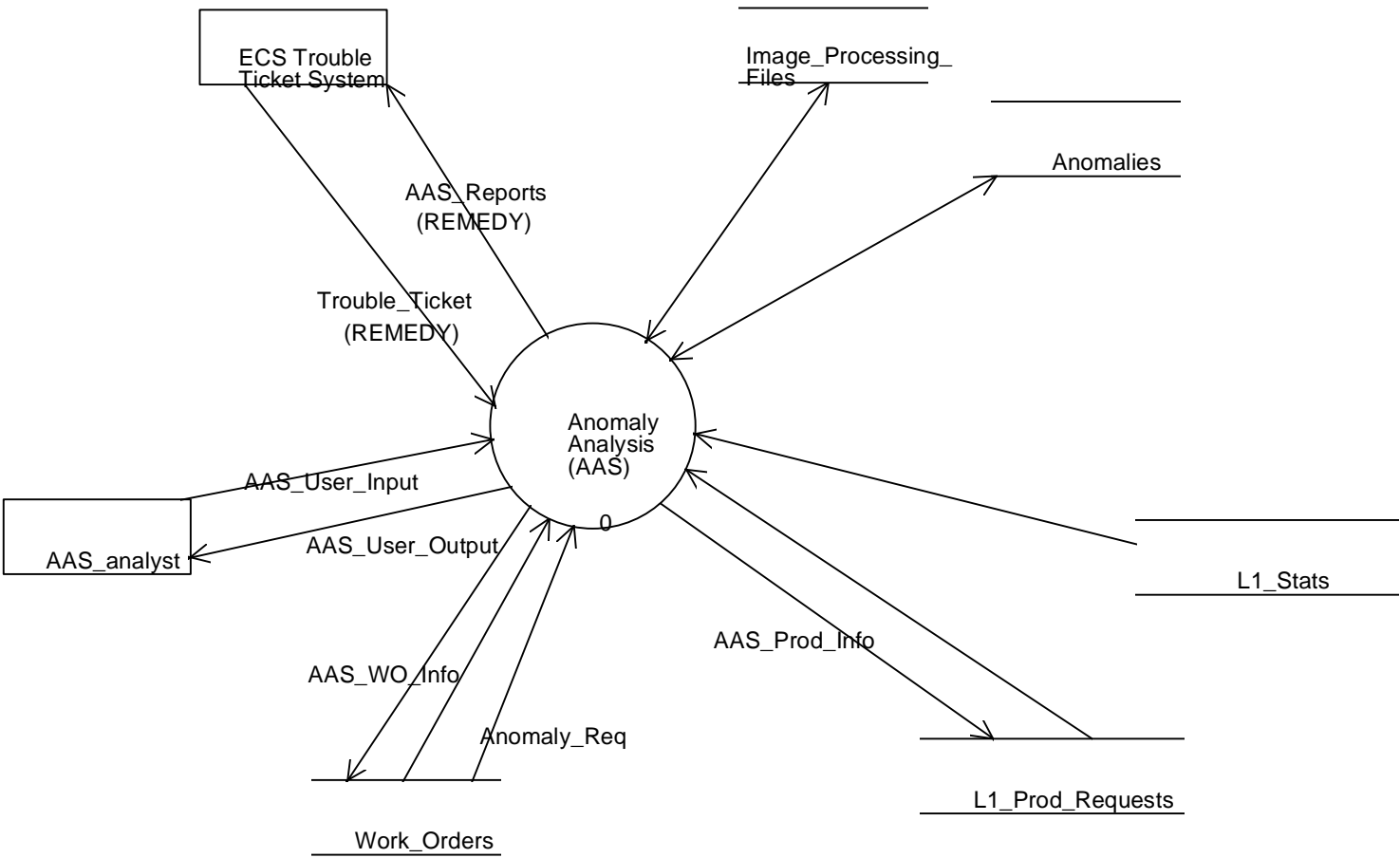


Figure 10-1. AAS Context Diagram

# REVIEW



**Figure 10-2. AAS Level 0 DFD**

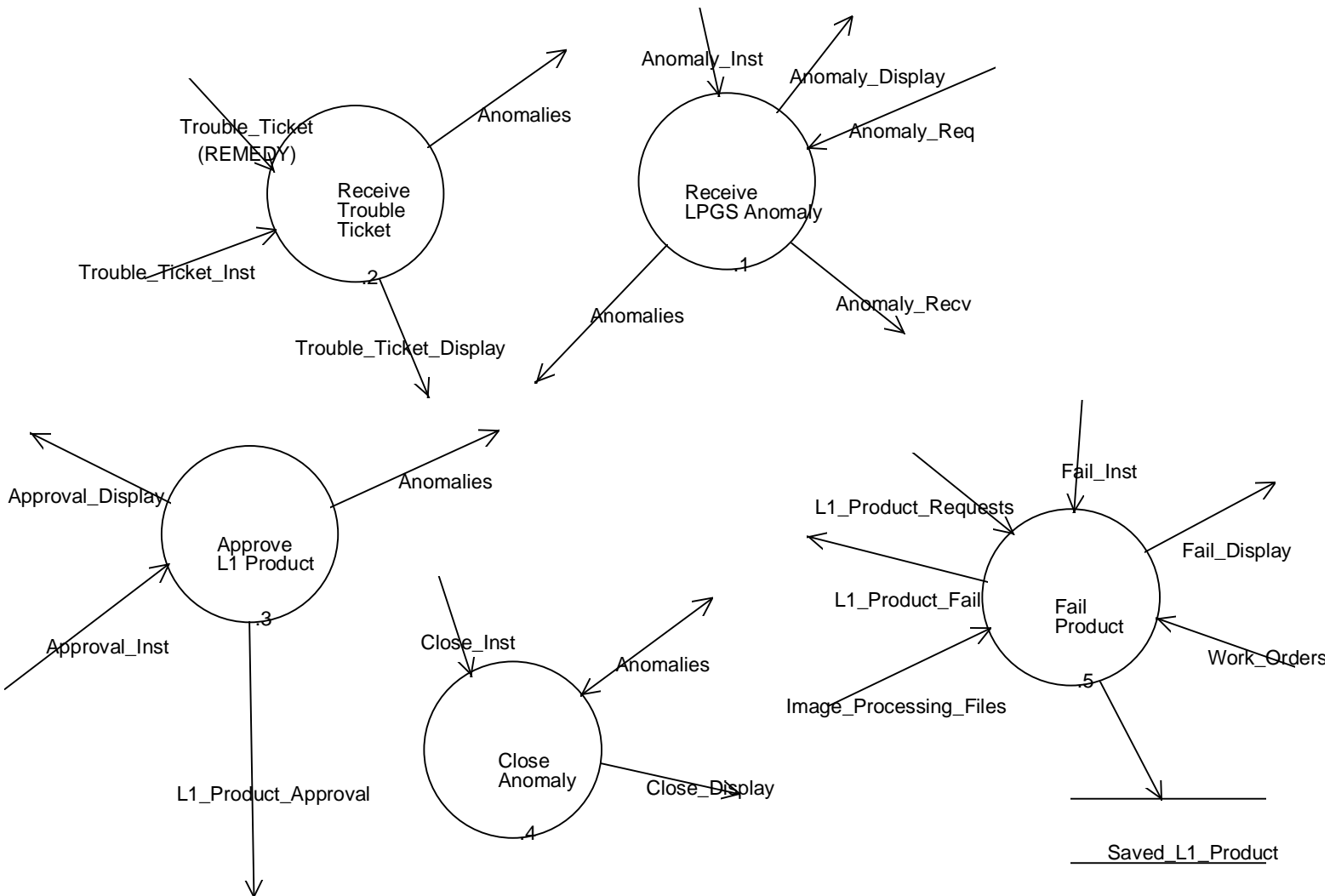


Figure 10-3. AAS DFD 1.0—Track Anomalies

## **REVIEW**

### **10.3.2.3 Approve L1 Product**

The Approve L1 Product function allows the analyst to manually approve an L1 product for distribution that was produced from a diagnostic work order. The analyst selects the work order to be approved. This function updates the work order state in the Work\_Orders table to indicate that the L1 product is ready to be shipped to the ECS.

### **10.3.2.4 Close Anomaly**

The Close Anomaly function allows the analyst to update the Anomalies table with the final resolution and/or analysis. When the anomaly is closed out, the entry in the Anomalies table becomes an historical entry. This information remains in the table for future reference by the analyst and for the anomaly history report.

### **10.3.2.5 Fail Product**

The Fail Product function allows the analyst to mark a product request as failed. The analyst uses this function when the analyst is unable to resolve the problem and the LPGS cannot produce the product. The analyst specifies the anomaly associated with product request to be marked as failed. The Anomalies table entry for the selected anomaly contains both the product request and current work order identifiers. The locations of the L0R input and the L1R/L1G output files are retrieved from the Product\_Requests and Work\_Orders tables, and all files associated with the request are copied to a tape. The Product\_Requests and Work\_Orders tables are updated to indicate that the processing has failed and that the associated files are eligible for deletion. After failing the product, the analyst uses the Transmit Anomalies function (see Section 10.3.6) to notify the ECS.

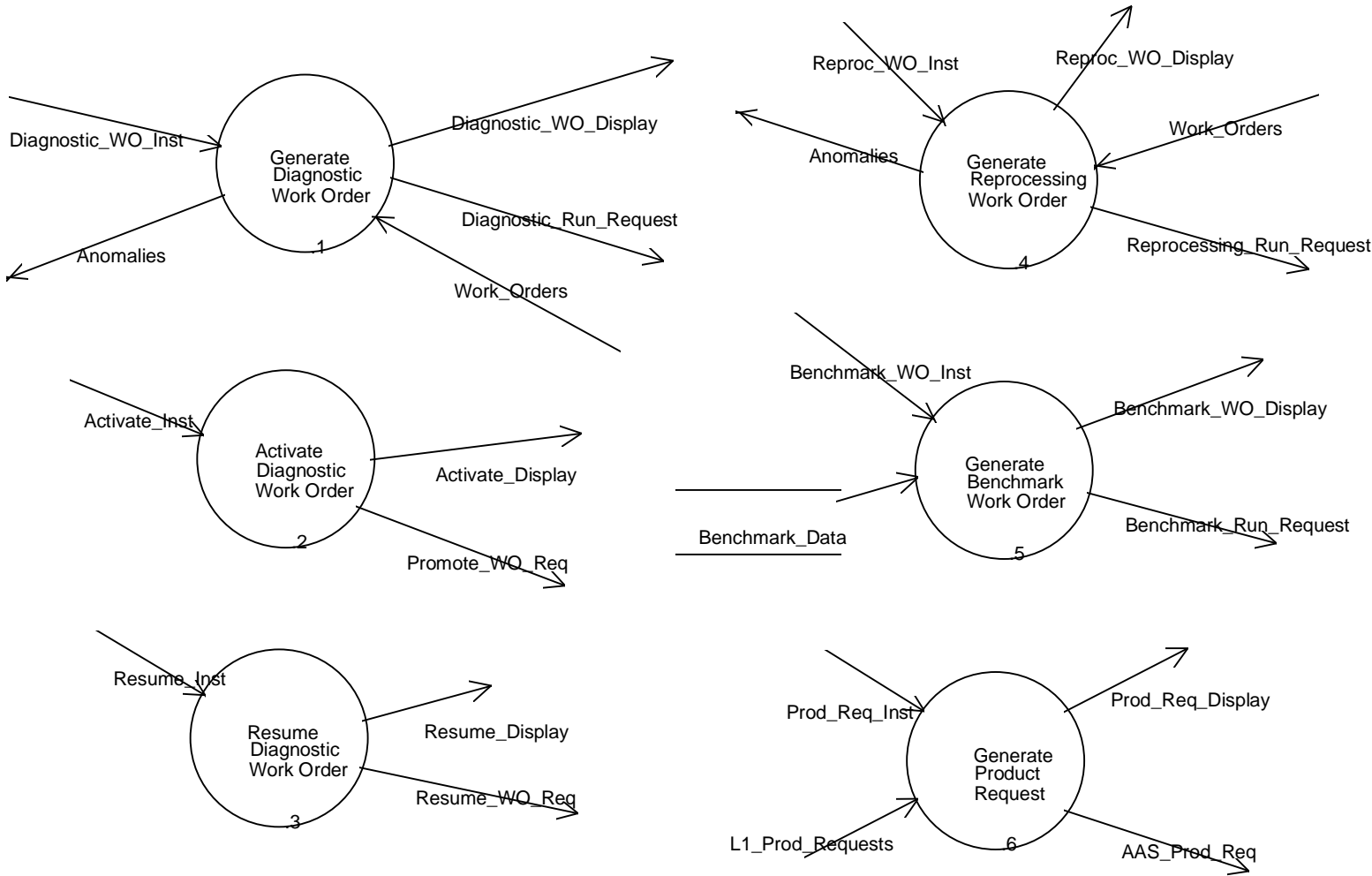
## **10.3.3 Perform Anomaly Runs**

The AAS includes functions that allow the analyst to initiate and control LPGS production processing for troubleshooting problems. Figure 10-4 depicts the Perform Anomaly Runs functions.

### **10.3.3.1 Generate Diagnostic Work Order**

The Generate Diagnostic Work Order function allows the analyst to generate a diagnostic work order based on an existing work order. The analyst selects the original work order to be used as the basis for the diagnostic work order. Then this function retrieves the work order information and displays it using Oracle Forms. The analyst can modify the work order by selecting additional pauses or different parameter values. When the analyst completes the updates, the work order is inserted into the database tables. A diagnostic work order is created in a suspend state to prevent the work order processing from starting when the analyst is not available. The Anomalies table is updated with the current work order ID.

Figure 10-4. AAS DFD 2.0—Perform Anomaly Runs



REVIEW

## **REVIEW**

### **10.3.3.2    Activate Diagnostic Work Order**

The Activate Diagnostic Work Order function allows the analyst to activate a diagnostic work order that was generated and placed in suspend mode by the Generate Diagnostic Work Order function. The analyst specifies the work order to be activated. This function updates the work order state in the Work\_Orders table to notify the PWS task that work order processing can begin.

### **10.3.3.3    Resume Diagnostic Work Order**

The Resume Diagnostic Work Order function allows the analyst to resume the processing of the work order after a halt in the script processing. The analyst selects the work order to be resumed. The work order state in the Work\_Orders table is updated to notify the PWS that the work order processing can resume.

### **10.3.3.4    Generate Reprocessing Work Order**

The Generate Reprocessing Work Order function allows the analyst to generate a reprocessing work order by creating a modified version of the original work order with any corrections that the analyst decides are needed to correct the problem. The analyst selects the original work order. This function retrieves the work order information and displays it using Oracle Forms. The analyst can modify the display with any necessary changes. When the analyst completes the updates, the work order is inserted into the database tables. A resulting reprocessing run would go through the system in the normal mode. The Anomalies table is updated with the current work order ID.

### **10.3.3.5    Generate Benchmark Work Order**

The Generate Benchmark Work Order function allows the analyst to generate a special work order using known good data to check out the LPGS. The analyst selects one of the standard benchmark data sets. This function can be used when a system-wide problem is suspected or when major changes are made to the LPGS configuration (hardware and/or software).

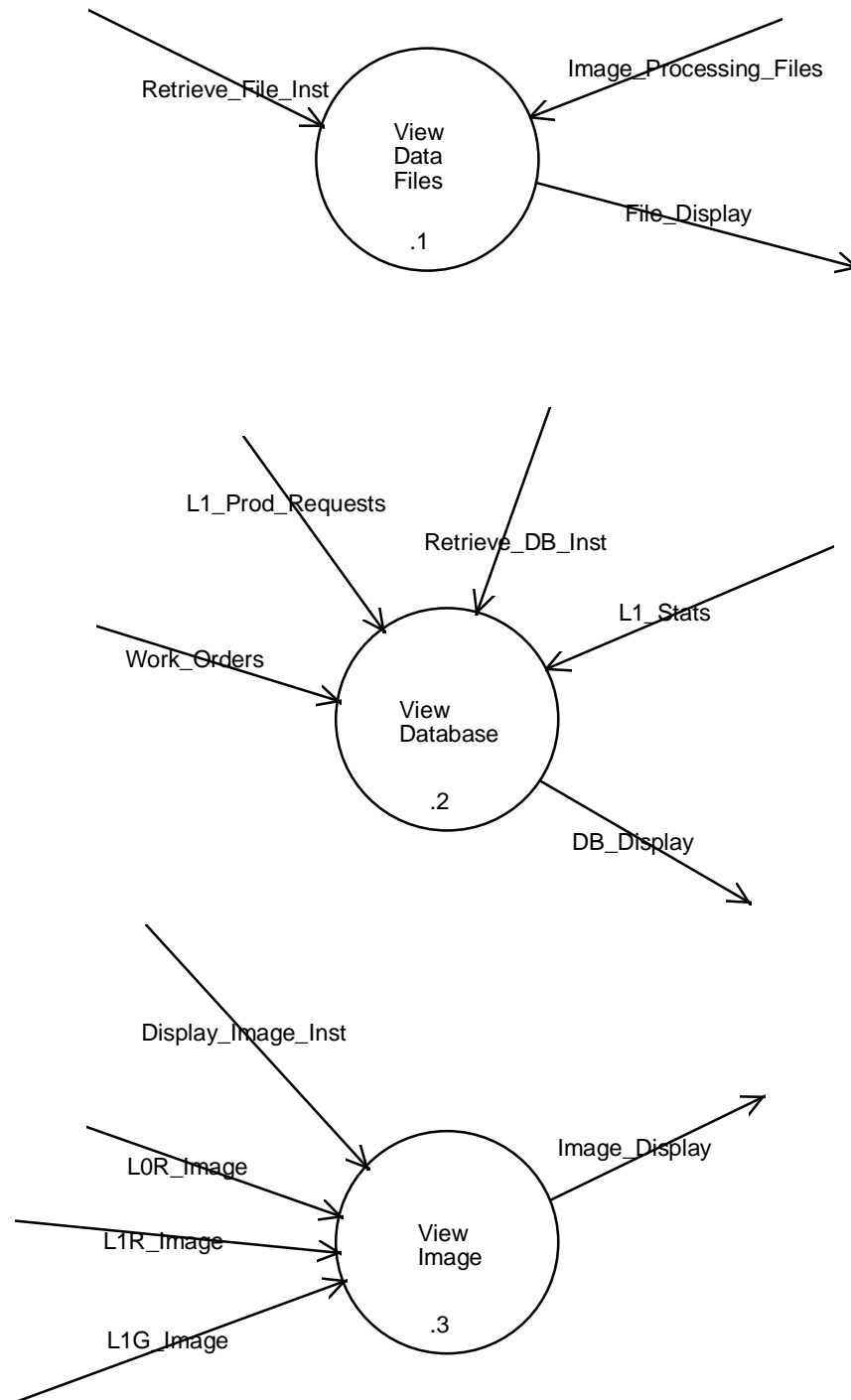
### **10.3.3.6    Generate Product Request**

The Generate Product Request function allows the analyst to manually generate a product request based on an existing request. This function is needed when the analyst wants to rerun processing for a trouble ticket. The analyst identifies the anomaly associated with the trouble ticket. This function retrieves the original request from the Product\_Requests table and displays the information. After the analyst indicates that all modifications have been made, the new product request is inserted into the Product\_Requests table.

## **10.3.4    View LPGS Data**

Figure 10-5 shows the Retrieve LPGS Data functions. This function allows the analyst, through the UI, to view any data files, database tables, and images associated with the anomalous work order to diagnose the problem.

## REVIEW



**Figure 10-5. AAS DFD 3.0—View LPGS Data**

## **REVIEW**

### **10.3.4.1 View Data Files**

The View Data Files function allows the analyst to view any of the text files (e.g., work order log or metadata) associated the product request being investigated. The analyst identifies the file to be viewed, and the file is displayed on the analyst's screen.

### **10.3.4.2 View Database Stores**

The View Database Stores function allows the analyst to view any of the pertinent database information related to the product request or work order. The analyst selects the information to be viewed. This function retrieves the information from the Product\_Requests, Work\_Orders, and other database tables and displays it on the analyst's screen.

### **10.3.4.3 View Image**

The View Image function allows the analyst to view or print any of the L0R, L1R, or L1G image files. The analyst selects the image, and the image is displayed on the analyst's screen. A COTS product will be used to support this function.

### **10.3.5 Generate History Report**

The Generate History Report function allows the analyst to generate a formatted display of historical information about anomalies. The analyst provides selection criteria through the UI. This function retrieves the applicable information from the Anomalies table and generates the anomaly history report.

### **10.3.6 Transmit Anomalies**

The Transmit Anomalies function allows the analyst to enter the response to a trouble ticket or add a new trouble ticket for internal problems outside the scope of resolution of LPGS into the ECS Trouble Ticket System using REMEDY.

# Section 11. Database Design

---

## 11.1 Introduction

The LPGS database provides communication between the subsystems for product requests, work order creation, processing, status, trending data, operational support, etc., for the LPGS.

During the preliminary design phase, a logical database design incorporated the functional data requirements and the interface between the database and the LPGS subsystems. During the detailed design phase, the logical design was converted to a physical design.

## 11.2 Logical Design

The goal of a logical design is to provide an accurate model of the information needs of the organization. This model, diagrammed as an entity relationship diagram (ERD), expresses the data in terms of entities, attributes, and relationships. The ERD also provides a model that is independent of any particular data storage and access method and allows objective decisions to be made about implementation techniques and coexistence with existing systems. The logical design is independent of any particular physical implementation. It is used to establish the physical database design.

Entity Relationship Modeling involves identifying the items of importance in an organization (entities), the properties of those items (attributes), and how they relate to one another (relationships). The LPGS Entity Relationship Model is expressed conceptually as an ERD (Figure 11-1). The following paragraphs describe these items in more detail.

### 11.2.1 Entities

Entities represent data items that play a functional role in the LPGS application and have their own set of attributes. An entity is a thing or object of significance about which information needs to be known or held. Each entity is expressed on the ERD as a box. The name of the entity is in bold followed by names of the attributes associated with the entity. Table 11-1 provides a description of each entity.

### 11.2.2 Attributes

Attributes are details that serve to express the state of, qualify, identify, classify, or quantify an entity. Each attribute on the ERD is defined as either a primary key (#) attribute, a mandatory (\*) attribute, or an optional (o) attribute. Primary keys are used to uniquely define an occurrence of the entity. Mandatory attributes require a data value at all times; optional attributes may be left null.

REVIEW

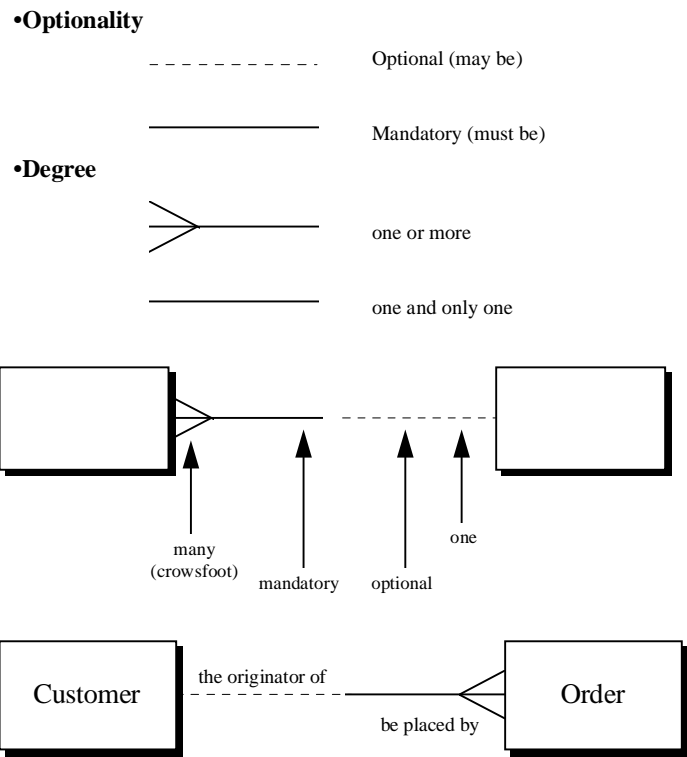


Figure 11-1. LPGS ERD

Table 11-1. Entities and Their Descriptions

TBS

## REVIEW

### 11.2.3 Relations

A relation is a significant association between two entities that is given a name. Relationships represent the association between the occurrences of one or more entities that are of interest to the LPGS. There is a cardinality associated with each relationship. The cardinality describes the number of occurrences of one entity that can be associated with each occurrence of the related entity. The following symbols depict relationships on the ERD.

For example,

- Each **Customer** may be the originator of one or more **Order**.
- Each **Order** must be placed by one and only one **Customer**.

### 11.3 Physical Design

The physical design of the database is the result of converting the logical design into a design that can be supported by the type of DBMS selected. Physical database design has several objectives:

- To optimize the logical schema as necessary while preserving the consistency and flexibility of data
- To determine the data storage requirements
- To determine data integrity and security rules
- To optimize the usage of the DBMS capabilities

A schema diagram (Figure 11-2) very similar to the ERD depicts the physical model of the design. The schema diagram represents tables, elements, indexes, integrity constraints, views, and snapshots. The following symbols depict tables and foreign keys on the schema diagram.

For example,

- Each **Customer** record must have a unique Customer\_Id.
- Each **Order** record would be validated against the **Customer** table using the foreign key Customer\_Id\_FK.

#### 11.3.1 Tables

The tables and columns compose the basic physical database design units. A database table is a basic unit of storage. Actual data are stored in rows according to a predefined table schema. Entities and relationships become tables, and attributes become table columns. Appendix C, the database table definition report, contains a full description of each table and its associated columns.

## REVIEW

### •Optionality

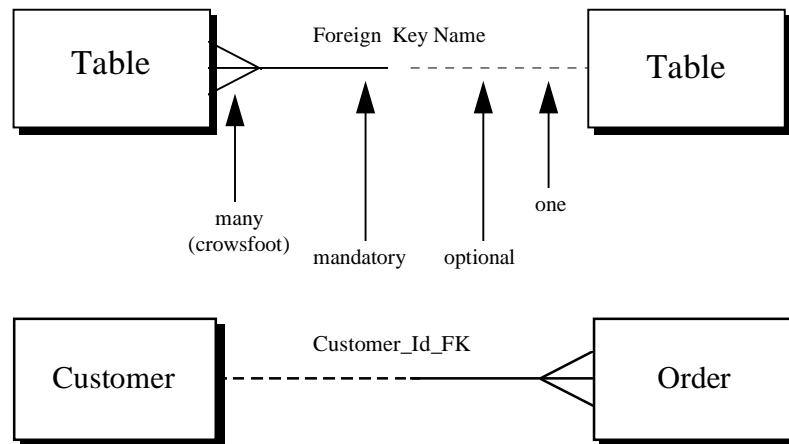
----- Optional (may be)

\_\_\_\_\_ Mandatory (must be)

### •Degree

> \_\_\_\_\_ one or more

\_\_\_\_\_ one and only one



**Figure 11-2. LPGS Schema Diagram**

### 11.3.2 Indexes

As part of the physical design analysis, indexes were determined for the tables. Indexes must be chosen carefully because they require maintenance as records are added and deleted from the tables and because they require additional storage space. Indexes can, however, significantly reduce the data retrieval time. Thus, indexes are crucial for performance. Indexes can also be used to ensure that data columns or combinations of columns are always unique within a given table.

### 11.3.3 Integrity Constraints

Integrity constraints provide a means of ensuring that changes made to the database do not result in a loss of data consistency. Thus, integrity constraints guard against accidental damage to the database. The following types of integrity constraints were considered in the design and are described in the database table definition report (Appendix C).

- Domain rules—Define a set of legal values (called domain rules) for a column or set of columns and instruct the DBMS to check all values that users try to store and to reject any that are illegal.

## REVIEW

- Unique keys—Prevent storing more than one record having duplicate values in a specified column or set of columns in a table. This is achieved by defining an implicit unique index on each set of columns by the DBMS. The DBMS will reject any attempt to store duplicate values that should be unique.
- Primary keys—Prevent storing more than one record having duplicate values in a specified column or set of columns in a table. It also disallows null values being entered in the specified columns. This constraint is enforced by the implicit creation of a unique index on that column and a not null constraint on that column by the DBMS.
- Foreign keys (referential constraints)—Whenever primary or alternate key values from one table are stored as a foreign key in another table, the reference must be valid. To preserve referential integrity, checking should be performed each time a record of a table containing foreign keys is inserted. The DBMS should check that the new foreign key values exist in another table and should not allow the insertion or update if they do not exist. Also, the DBMS performs a check every time a record or a table is deleted or modified because records of other tables may exist that point to that table row using foreign keys.
- Null—A rule defined on a single column that allows or disallows inserts or updates of rows containing a null for the column.

### 11.3.4 Database Storage Requirements

During the detailed design phase, the rows for each table are defined along with their minimum and maximum sizes. Using assumptions from the application, the total number of rows for each table is estimated. These estimations are fed into formulas to calculate the starting and maximum size of each table and the size of its associated primary key. Tables and indexes are allocated to tablespaces, and tablespaces are combined to estimate the total database size for the application. The LPGS database is divided into three tablespaces for data:

- Static data—Tables allocated to this table space seldom change. Nominal statistics, system parameters, directives, etc., are examples of tables allocated to this tablespace.
- Dynamic data—Tables allocated to this tablespace change often during processing. These tables support ingest, product request and work order definition, processing parameters, control, etc.
- Trending data—Data in this tablespace hold the trending output from the LPGS that is later transferred to the IAS.

For the static data, the number of rows is a relative constant. The storage needed for the static data tables is the sum of each table's row length times the number of rows in the tables. The row length includes DBMS overhead for each column and row in the tables. In addition, the DBMS organizes the data into blocks requiring additional table overhead. For the dynamic and trending data tables, the storage is calculated based on the maximum number of rows expected at a time.

## REVIEW

### 11.4 Database Sizing

This section presents the assumptions for database sizing, estimates on table sizing, and estimated database size.

Table 11-2 lists the assumptions used in calculating the size of the database. Table 11-3 lists the table sizes and row estimates. Table 11-4 provides the details of the results calculated for the database tables and rows over the life of the system.

### 11.5 Database Administration

This section briefly describes database administration procedures to facilitate data security, integrity, backup, and recovery. A detailed description of the procedures will be described in the LPGS database administration manual.

#### 11.5.1 Security

Database security will be accomplished at two levels, system security and data security. Oracle-provided security features will be used to control the LPGS database accessibility and usage.

System security includes the mechanisms that control the access and use of the database at the system level. The database system security options include

- Entry of valid user name and password combination
- Authorization to connect to the database
- Amount of disk space available to the objects of a user/process
- Limits of resources for a user/process
- Audit of designated limits for a user/process
- System operations available for performance by a user group

Data security includes the mechanisms that control the access and use of the database at the object level. The database data security options include

- Identification of users that have access to a specific schema object and the specific types of actions allowed on the object for each user (for example, user A can issue SELECT and INSERT statement but not DELETE statements using the B table)
- Actions, if any, that are stored in the LPGS data dictionary

## REVIEW

**Table 11-2. LPGS Database Sizing Assumptions**

Procedures/Programs		Calibration	
1	Procedures/Work Order	365	PASC/Year
24	Procedures	8	FASC/Year
36	Scripts	20	GLC/Year
24	Programs	0.5	Probability of Specifying Gain
1.5	Programs/Script	1	Gain/Band
7	Scripts/Procedure		
6	Parameters/Program	<b>Duration</b>	
		20	Workdays/Month
<b>Scenes</b>		12	Months/Year
25	Scenes/Day	5	Years
1	Work Orders/Scene	4	Quarters/Year
1	Scenes/Work Order		
9	Bands/Scene	<b>Miscellaneous</b>	
1	LOR Products/Scene	8	File Types
6	Major frames/Scene	1.5	Default Directory/File
1.11	Calibration Source/Scene	1	L1RG Products/Work Order
375	Scans/Band	2	Events/Program execution
		0.5	Orbit Based Request/CSR
<b>Trending</b>		0.5	WRS Based Request/CSR
4	Regions of interest/ Detector/Scan	4	CPFs generated/Year
2	Scan Directions	12	Quarterly Reports
2	SCS Levels	3	Monthly Reports
3	Segments/Detector/Scan		
375	Scan lines/Detector/Scene	<b>Ephemeris</b>	
6600	Pixels/Detector/Scan	1	Ephemeris Request/month
0.0000001	Pixel of interest/ Detector/Scan	2	Types of ephemeris each received once/month
16	Detectors/Band		
		<b>Product Requests</b>	
<b>Number of Scenes Used for</b>		1	Product Request/Work Order
2	Band to Band/Month	1.1	Work Order/Product Request
2	Band alignment/Month	0.1	Anomalies/Product Request
1	Image to Image/Month	1	Customer/Product Request
2	Geodetic/Week		
1	Geometric/Month	<b>Work Order/Anomalies</b>	
1	Mirror/Month	1	Anomaly/Work Order
2	Sensor Alignment/Week	3	Work Order/Anomaly

## REVIEW

**Table 11-3. Table Size and Row Estimates (1 of 2)**

**NOTE:** The following sizing estimates for trending data include Geometry trending data only and do not include Radiometry. New sizing estimates will be generated after the final trending attributes have been incorporated into the design.

Table Name	Table Space	Number of Rows	Row Size Bytes	Total Size
Geometric_Stats	Trend_Data			
I2I_Stats	Trend_Data			
Mirror_Stats	Trend_Data			
Geodetic_Stats	Trend_Data			
S_Align_Stats	Trend_Data			
B_Align_Stats	Trend_Data			
Scene	Trend_Data			
Proc_Scene	Trend_Data			
PCD_Major_Frame	Trend_Data			
Scan_Group	Trend_Data			
Band	Trend_Data			
Pixel	Trend_Data			
Facet_Aggregates	Trend_Data			
Detector	Trend_Data			
Det_Aggregates	Trend_Data			
Facets	Trend_Data			
Segment	Trend_Data			
Scan	Trend_Data			
Region	Trend_Data			
Scan_Line	Trend_Data			
Nominal_L0R_stats	Static_Data			
Sys_Parms	Static_Data			
Foreign_Host	Static_Data			
Def_Directory	Static_Data			
Module	Static_Data			
Message	Static_Data			
Sub_Module	Static_Data			
Def_Parm	Static_Data			
Anomalies	Dynamic_Data			
CPF_Catalog	Dynamic_Data			
Customer_Information	Dynamic_Data			
Ephem_Requests	Dynamic_Data			
Ephem_Files	Dynamic_Data			

## REVIEW

**Table 11-3. Table Size and Row Estimates (2 of 2)**

Table Name	Table Space	Number of Rows	Row Size Bytes	Total Size
CSR_Orbits	Dynamic_Data			
CSR_WSR	Dynamic_Data			
IPC_Directive	Dynamic_Data			
Cal_Scene_Request	Dynamic_Data			
File_Transfer	Dynamic_Data			
CSR_Gains	Dynamic_Data			
L0R_Data_Datalog	Dynamic_Data			
L1R_Data_Catalog	Dynamic_Data			
Processing_Stats	Dynamic_Data			
Product_Request	Dynamic_Data			
Work_Order	Dynamic_Data			
Event	Dynamic_Data			
QAS_Results	Dynamic_Data			
WO_Script	Dynamic_Data			
WO_Param	Dynamic_Data			

## REVIEW

**Table 11-4. Estimated Database Size Summary**

**NOTE:** The following sizing estimates for trending data include Geometry trending data only and do not include Radiometry. New sizing estimates will be generated after the final trending attributes have been incorporated into the design.

Initial Size: xxx.xx Kb                      End Size: yyy.yy Gb                      Max. Size: zzz.zz Gb

Blocksize: 2048

No. Tablespaces: 6                      No tables: 48                      No keys: xx

Space needed for tables:

Tablespace	Initial	End Size	Max. Size
Static-Data			
Dynamic-Data			
Trending			
Total			

Space needed for indexes:

Tablespace	Initial	End Size	Max. Size
Static-Data			
Dynamic-Data			
Trending			
Total			

## **REVIEW**

### **11.5.2 Backup**

Oracle-provides the capability to perform both full and partial backups of the database. A full backup is defined as including all tables and files associated with the LPGS database system. The database must be closed and offline to perform a full backup, and thus this should be a scheduled activity.

A partial backup, sometime referred to as an incremental backup, can occur while the database is online and active. Only specific data are processed in the backup. This activity permits the storage of selected database tables, while not impacting the current operations. This procedure should enhance but not replace scheduled full backups.

### **11.5.3 Recovery**

Recovery after a crash (power failure) is automatically handled by Oracle as part of the normal database startup procedure. All transactions committed before the system crash are recovered, and the uncommitted transactions are rolled back.

In the event of a media failure (a disk crash or corruption), the data on that device can be restored from a backup copy. The data will be restored to the time of the backup (Oracle Export utility can be used); data modifications that were applied after the backup will be reapplied if the log files are available.

# Section 12. User Interface

---

## 12.1 Introduction

This section describes the user interface for both the LPGS operator and analyst. Additional details regarding the user interface can be found in the LPGS User's Guide (Reference 13).

The operator monitors the LPGS processing status as necessary. The operator user interface (OUI) is the primary interface between the operator and the LPGS subsystems, including the DMS and the PCS. This interface uses Oracle Forms to provide a menu-driven user interface.

The analyst performs visual assessments of L1 digital images and investigates the source of internal processing anomalies. The analyst user interface (AUI) is the primary interface between the analyst and the QAS and the AAS. This interface uses Oracle Forms to provide a menu-driven environment for evaluation and assessment functions.

### 12.1.1 Design Considerations

The OUI uses a set of standalone COTS applications that are compatible but loosely integrated. For instance, the user will use Oracle Forms displays to monitor the product request processing status.

The AUI uses Oracle Forms to provide menu access to custom COTS software needed for visual quality evaluation and anomaly analysis. It also uses Oracle Forms to perform functions on work orders and product requests associated with anomalies.

The screens in the OUI and AUI will employ the same look and feel to minimize training in the cases where the operator does analyst functions or visa versa.

### 12.1.2 Design Assumptions and Open Issues

The design assumptions for the LPGS user interface are as follows:

- The OUI runs on an X terminal connected via Ethernet to an SGI Origin 2000
- The AUI runs on an SGI 02 workstation connected via FDDI to the LPGS
- The same person may execute both the OUI and the AUI
- Only users who have been provided access to Oracle Forms can run the interface (no outside users)
- The OUI and the AUI use a common menu interface

### 12.1.3 Design/Development Tools

This subsection presents the tools used to design and develop the user interface for the operators and analysts. It is anticipated that the AUI will use some of the same tools as the OUI, whereas the OUI will not need to use any of the AUI image visualization tools.

## **REVIEW**

### **12.1.3.1 Common Tools**

Oracle Corporation's Cooperative Development Environment (CDE) tools were used for designing the OUI and the AUI. CDE tools are built on top of a layer called the Oracle Toolkit. The Oracle Toolkit makes it possible to create platform-independent applications. These applications can be developed in one environment (i.e., platform/GUI protocol combination) and seamlessly transferred to another environment. Oracle Forms was the CDE tool used to design the LPGS interface.

#### **12.1.3.1.1 Oracle Forms**

Oracle Forms is an integrated set of modules that assist in the development of Oracle applications. Oracle Forms consists of modules for building both forms and menus. The Forms module provides the capability to build and run screen-oriented applications for data entry, retrieval, and display. The Menus module provides facilities for building custom menus that can be attached to a form. By combining forms and menus, menu-driven applications can be built that execute multiple data processing tasks via a common interface.

#### **12.1.3.2 Analyst Tools**

The Oracle Forms and a COTS image viewing product were used to design the AUI. Oracle Forms provides a basic set of menus to access the image analysis functions.

## **12.2 User Interface Functions**

The LPGS user interface is a menu-driven system that provides the operator access to system functions, monitor functions, product request functions, anomaly analysis functions, and quality assessment functions.

### **12.2.1 System Functions**

The system functions include the recovery, manage resources, configure LPGS, start tasks, restart tasks, shutdown tasks, and exit operator functions.

#### **12.2.1.1 Recovery**

The Recovery function provides the operator with the capability to clean up product requests and allows the operator to mark data files associated with specific product requests or work orders for deletion.

#### **12.2.1.2 Manage Resources**

This group of functions allows the operator to manage the LPGS system resources. The View Disk Storage capability allows the operator to view the available disk storage. The Delete Trending capability allows the operator to delete the trending data associated with a selected product request. The Delete Image Files allows the operator to delete the image files associated with a selected product request.

## **REVIEW**

### **12.2.1.3 Configure LPGS**

The Configure LPGS group of functions allows the operator to modify parameters used to control LPGS processing. The System Params function allows the operator to modify the system parameters that effect how the LPGS system operates. The ECS Interface Params function allows the operator to modify the parameters associated with the LPGS/ECS interface.

### **12.2.1.4 Start Tasks**

The Start Tasks function provides the operator with the capability to start the LPGS background tasks.

### **12.2.1.5 Restart Tasks**

The Restart Tasks function allows the operator to restart any of the LPGS background tasks. The restart function lets the operator restart all of the background tasks or individual tasks.

### **12.2.1.6 Shut Down Tasks**

The Shut Down Tasks function allows the operator to terminate the LPGS application software. Shut Down allows a graceful or immediate shutdown. During a graceful shutdown, current processing finishes, files are closed, the application disconnects from the database, etc. During an immediate shutdown, all processing terminates without delay. An immediate shutdown is similar to an abort and would normally be used only in emergency situations.

### **12.2.1.7 Exit**

The Exit function allows the operator to exit the LPGS user interface.

## **12.2.2 Monitor Functions**

The monitor functions include the database function, view event log, and accounting report operator functions.

### **12.2.2.1 Database**

The database function allows the operator to access the LPGS database using SQL\*Plus commands. Using SQL\*Plus, the operator can perform ad hoc queries of the database.

### **12.2.2.2 View Event Log**

The View Event Log function provides the operator with a display of the LPGS event log. The operator can filter the displayed output by time, severity, product request, work order, script, or program.

## **REVIEW**

### **12.2.2.3 Processing Statistics Report**

The Processing Statistics Report function allows the operator to display information concerning LPGS processing that is stored in the database. The operator selects to view the accounting statistics for a given time range.

### **12.2.3 Product Request Functions**

The product request functions include the promote product request, LOR receipt, product request information, cancel product request, and work order operator functions.

#### **12.2.3.1 Promote Product Request**

Promote Product Request allows the operator to promote a product request. The promotion request is received from the ECS via phone or e-mail.

#### **12.2.3.2 LOR Receipt**

The operator can ingest L0 data via a back door mechanism, such as tape or ftp. LOR Receipt allows the operator to notify the LPGS that LOR data are available for processing.

#### **12.2.3.3 Product Request Information**

Product Request Information allows the operator to view product request information stored in the LPGS database. The information displayed includes the status of the product request, as well as the status of any work orders associated with the product request.

#### **12.2.3.4 Cancel Product Request**

Cancel Product Request provides the operator with the capability to cancel a product request based on a request from the ECS via phone or e-mail.

#### **12.2.3.5 Work Order**

The work order group of commands allows the operator to resume, pause, promote, cancel, and display the work orders. Resume work order allows the operator to restart the processing of a paused work order. This is usually done by the operator after the visual inspection of an image is complete. Pause work order allows the operator to clear or set pauses between the execution of all or selective scripts being run in support of the work order. Promote work order allows the operator to promote the work order so that it runs next. Cancel work order allows the operator to terminate the processing of a work order. Display work order allows the operator to view the work order information stored in the LPGS database.

### **12.2.4 Anomaly Analysis Functions**

The anomaly analysis functions include the view, work order, anomaly, and trouble ticket groups of analyst functions.

## **REVIEW**

### **12.2.4.1 View**

The view group of functions allows the analyst to display images, files, event log, quality results, and product request information. View Images provides the analyst with the capability to display the L0R, L1R, and L1G images. View Files allows the analyst to view the contents of ASCII files. View Event Log provides the analyst with a display of the LPGS event log. View Quality Results allows the analyst to view the L1R and L1G automated quality assessment results that are stored in the LPGS database. View Product Request provides the analyst with a method to view the product request information which is stored in the LPGS database.

### **12.2.4.2 Work Order**

The work order group of functions allows the analyst to generate, modify, resume, approve distribution, activate, fail product, cancel, and display the work orders. Generate work order allows the analyst to manually enter a work order into the LPGS system to investigate or resolve an anomaly. Modify work order allows the analyst to modify the pauses or program parameters associated with specific scripts in AAS-generated work orders. Resume work order provides the analyst with a method to restart a halted work order. Approve Distribution allows the analyst to approve the distribution of successfully generated L1 products. Activate work order provides the analyst with a method to activate an inactive work order. Fail product allows the analyst to fail a work order that cannot be resolved. Cancel work order provides the analyst with a means to cancel an AAS-generated work order. The analyst can elect to cancel the work order immediately or at the end of the currently executing script. Work Order Information provides the analyst with a method to view the information associated with a work order that is stored in the LPGS database.

### **12.2.4.3 Anomaly**

The anomaly group of functions allows the analyst to insert information, edit and display information, view anomaly report, and view history report. Insert information allows the analyst to log new anomalies into the LPGS. Edit/Display information allows the analyst to view and modify anomalies previously entered into the LPGS system. Anomaly Report provides the analyst with the capability to filter and display the anomalies recorded in the LPGS.

### **12.2.4.4 Trouble Ticket**

The trouble ticket group of functions allows the analyst to display a trouble ticket, create a product request, and generate response to a trouble ticket. Display a trouble ticket provides the analyst with the capability to display the trouble tickets recorded in the ECS REMEDY system. Upon receipt of a trouble ticket that indicates a problem with a previously delivered L1 product, the analyst must generate a new product request so the problem can be evaluated. Create Product Request allows the analyst to generate the new product request based on the original product request. Generate Response provides the analyst with a method to provide a response to the ECS concerning the trouble ticket.

## **REVIEW**

### **12.2.5 Quality Assessment Functions**

The quality assessment functions include the view and approve/reject groups of analyst functions.

#### **12.2.5.1 View**

The view group of functions allows the analyst to display images and results. View Images provides the analyst with the capability to display the L0R, L1R, and L1G images. View Results allows the operator to view the L1R and L1G quality results, which are stored in the LPGS database.

#### **12.2.5.2 Approve/Reject**

Approve/Reject provides the analyst with the capability to approve or reject the image after the visual inspection is completed.

## REVIEW

### Appendix A. LPGS Requirements Traceability Matrix

Rqmt. No.	Requirement Summary	PCS	DMS	RPS	GPS	QAS	AAS	UI	GL	HW	OPS
3.1	System-Level Requirements										
3.1.1	The LPGS shall nominally generate L1 data products on a FIFO basis.	X	X							X	
3.1.2	The LPGS shall provide the capability to move a Level 1 image processing work order within the FIFO queues according to operator direction.							X			X
3.1.3	Deleted										
3.1.4	Deleted										
3.1.5	The LPGS shall provide the capability to generate and report LPGS error messages.	X	X	X	X	X	X	X	X		
3.1.6	The LPGS shall provide an interactive capability to facilitate detection and correction of abnormal system conditions.	X	X	X	X	X	X	X	X	X	X
3.1.7	The LPGS shall provide the capability to isolate system faults.	X					X			X	X
3.1.8	The LPGS shall provide the capability to recover from system faults.	X								X	X
3.1.9	The LPGS shall provide the capability to test LPGS functions and external interfaces.	X	X	X	X	X	X	X	X	X	X
3.1.10	The LPGS shall provide the capability to support attended operations 24 hours per day, 7 days per week, on a continuous basis.	X	X	X	X	X	X	X	X	X	X
3.1.11	The LPGS shall provide the capability to support unattended, automatic processing 16 hours per day, 7 days per week, on a continuous basis.	X	X	X	X	X			X	X	

## REVIEW

Rqmt. No.	Requirement Summary	PCS	DMS	RPS	GPS	QAS	AAS	UI	GL	HW	OPS
3.1.12	The LPGS shall provide the capability to support Landsat 7 operations for a minimum mission life of 5 years.	X	X	X	X	X	X	X	X	X	X
3.1.13	The LPGS shall provide the capability to execute diagnostic tests for verifying proper operation of system capabilities and components.	X	X	X	X	X	X	X	X	X	X
3.1.14	The LPGS shall provide the capability to support end-to-end testing of Level 1 processing functions.	X	X	X	X	X	X	X	X	X	X
3.1.15	The LPGS shall provide the capability to control LPGS operations.		X					X			X
3.1.16	The LPGS shall provide the capability to monitor LPGS operations.		X				X	X			X
3.1.17	The LPGS shall provide the capability to reconfigure LPGS system resources.							X		X	X
3.1.18	The LPGS shall provide the capability to support software upgrades while supporting normal operations.	X	X	X	X	X	X	X	X	X	X
3.1.19	The LPGS shall be capable of making all software and databases used in operations accessible to ECS for archiving.		X							X	X
3.1.20	The LPGS design shall be scaleable to allow for future growth in processing capability.	X	X	X	X	X	X	X	X	X	
3.1.21	The LPGS shall be able to generate level 1 digital images corresponding to either heritage world-wide reference system (WRS) scenes or to a partial ETM+ subinterval up to an area equivalent to three WRS scenes.			X	X						
3.1.22	The LPGS shall be capable of recovering from failures and aborts in a controlled manner.	X	X	X	X	X	X	X	X	X	

## REVIEW

Rqmt. No.	Requirement Summary	PCS	DMS	RPS	GPS	QAS	AAS	UI	GL	HW	OPS
3.2	External Interface Requirements										
3.2.1	The LPGS shall interface with ECS to receive the following:										
3.2.1.1	LOR files (includes associated PCD files, MSCD files, and CPFs)		X							X	
3.2.1.2	Level 1 image processing requests		X							X	
3.2.1.3	Data availability notification		X							X	
3.2.1.4	Production status requests <b>NOTE:</b> This req to be deleted										
3.2.1.5	Product cancellation requests <b>NOTE:</b> This req to be deleted										
3.2.1.6	Product problem report (trouble ticket)		X				X			X	X
3.2.2	The LPGS shall interface with ECS to coordinate transfer of the following:										
3.2.2.1	LPGS L1 digital images		X							X	
3.2.2.2	Processing status <b>NOTE:</b> This req to be deleted										
3.2.2.3	Production quality and accounting information <b>NOTE:</b> This req to be deleted										
3.2.2.4	L1 processing statistics <b>NOTE:</b> This req to be deleted										
3.2.2.5	L1 metadata		X							X	
3.2.2.6	PCD file (consensus)		X							X	
3.2.2.7	MSCD file (consensus)		X							X	
3.2.2.8	IC data file		X							X	
3.2.2.9	CPF		X							X	
3.2.2.10	Geolocation table		X							X	
3.2.3	The LPGS shall interface with the IAS to provide Level 1 radiometric characterization data.		X	X						X	
3.2.4	The LPGS shall interface with DHF to provide L1 processing anomaly reports. <b>NOTE:</b> This req to be deleted										
3.3	Functional Requirements										
3.3.1	Retrieve LOR files.										

## REVIEW

Rqmt. No.	Requirement Summary	PCS	DMS	RPS	GPS	QAS	AAS	UI	GL	HW	OPS
3.3.1.1	The LPGS shall provide the capability to receive LOR data inputs from ECS. This data shall include the following items:		X							X	
3.3.1.1.1	Level 1 image processing request that includes the following:										
3.3.1.1.1a	Selected coordinate reference system for map projection		X							X	
3.3.1.1.1b	Requested orientation (Nominal Path or North Up)		X							X	
3.3.1.1.1c	Variable grid cell size selection		X							X	
3.3.1.1.1d	Output format selection		X							X	
3.3.1.1.1e	Resampling filter		X							X	
3.3.1.1.1f	Selected bands		X							X	
3.3.1.1.1g	Selected scene or subinterval identification		X							X	
3.3.1.1.1h	L1R or L1G image processing selection		X							X	
3.3.1.1.1i	Geographic area		X							X	
3.3.1.1.1j	WRS (path/row) scene identifier		X							X	
3.3.1.1.1k	IC or CPF (default = CPF)		X							X	
3.3.1.1.2	Data availability notification specifying the location of associated LOR product files ready for retrieval		X							X	
3.3.1.1.3	LOR product files (includes LOR image data, PCD, MSCD, and associated calibration files), IC, and calibration parameter		X							X	
3.3.1.1.4	Production status request <b>NOTE:</b> This req to be deleted										
3.3.1.1.5	Product cancellation request <b>NOTE:</b> This req to be deleted										
3.3.1.1.6	Product problem report (trouble ticket)		X				X			X	X
3.3.1.2	The LPGS shall provide the capability to create and send LOR product requests to ECS.		X							X	

## REVIEW

Rqmt. No.	Requirement Summary	PCS	DMS	RPS	GPS	QAS	AAS	UI	GL	HW	OPS
3.3.1.3	The LPGS shall coordinate resolution of data transfer problems with any LOR product file with ECS.									X	X
3.3.1.3.1	The LPGS shall be able to detect data transfer problems.		X								
3.3.1.3.2	The LPGS shall be able to re-retrieve data.		X							X	
3.3.2	The LPGS shall be able to extract and process Landsat 7 ETM+ Earth image data from the LOR Earth image data file to produce radiometrically corrected L1R digital images.			X							
3.3.2.1	The LPGS shall be able to extract and process attitude, and ephemeris data from the LOR PCD files.		X	X							
3.3.2.2	The LPGS shall be able to extract parameters from the LOR IC or CPF for use in L1R and L1G processing.		X	X							
3.3.2.3	The LPGS shall be able to generate gains and biases from either IC data or from the CPF. The default shall be the CPF.			X							
3.3.2.4	The LPGS shall be able to extract and process mirror scan correction coefficients from the LOR MSCD file to determine scan line quality.		X	X							
3.3.2.5	The LPGS shall be capable of detecting the following image artifacts:										
3.3.2.5.1	Striping		X	X							
3.3.2.5.2	Banding		X	X							
3.3.2.5.3	Coherent noise		X	X							
3.3.2.5.4	Deleted										
3.3.2.5.5	Scan-correlated shift		X	X							
3.3.2.5.6	Saturated detectors		X	X							
3.3.2.5.7	Dropped scan lines		X	X							
3.3.2.6	The LPGS shall be capable of characterizing the following image artifacts:										
3.3.2.6.1	Striping			X							

## REVIEW

Rqmt. No.	Requirement Summary	PCS	DMS	RPS	GPS	QAS	AAS	UI	GL	HW	OPS
3.3.2.6.2	Banding			X							
3.3.2.6.3	Coherent noise			X							
3.3.2.6.4	Deleted										
3.3.2.6.5	Deleted										
3.3.2.6.6	Saturated detectors			X							
3.3.2.6.7	Dropped scan lines			X							
3.3.2.7	The LPGS shall be capable of applying compensation for the following image artifacts:										
3.3.2.7.1	Striping			X							
3.3.2.7.2	Banding			X							
3.3.2.7.3	Coherent noise			X							
3.3.2.7.4	Memory effect			X							
3.3.2.7.5	Scan correlated shift			X							
3.3.2.7.6	Inoperable detectors			X							
3.3.2.7.7	Saturated detectors			X							
3.3.2.7.8	Dropped scan lines			X							
3.3.2.8	The LPGS shall be capable of applying compensation for gain changes within a requested L1 scene or subinterval as identified in the L0R metadata.			X							
3.3.2.9	The LPGS shall be capable of producing L1R data from L0R data for both the ascending and descending portions of the Landsat 7 orbit.			X							
3.3.2.10	The LPGS shall be able to produce L1R digital images for any combination of the eight spectral channels.			X							
3.3.2.11	The LPGS shall assemble and append to the L1R digital images all of the applicable metadata and quality and accounting data gathered in the construction of the L1R digital image. The complete L1R digital image package contains the following data elements as a minimum:										
3.3.2.11.1	Level 1R digital image (all requested bands)		X								
3.3.2.11.2	L1 metadata file		X								

## REVIEW

Rqmt. No.	Requirement Summary	PCS	DMS	RPS	GPS	QAS	AAS	UI	GL	HW	OPS
3.3.2.11.3	DELETED										
3.3.3	The LPGS shall be able to extract and process Landsat 7 ETM+ Earth image data from the L1R Earth image data files to produce systematically corrected L1G digital images.				X						
3.3.3.1	The LPGS shall have the capability to resample L1R digital images and apply the following map projections:				X						
3.3.3.1.1	Space oblique Mercator				X						
3.3.3.1.2	Universal Transverse Mercator (UTM)				X						
3.3.3.1.3	Lambert conformal conic				X						
3.3.3.1.4	Transverse Mercator				X						
3.3.3.1.5	Oblique Mercator				X						
3.3.3.1.6	Polyconic				X						
3.3.3.1.7	Polar stereographic				X						
3.3.3.2	The LPGS shall support the following compensation resampling methods:										
3.3.3.2.1	Nearest neighbor				X						
3.3.3.2.2	Cubic convolution				X						
3.3.3.2.3	Modulation Transfer Function (MTF)				X						
3.3.3.3	The LPGS shall have the capability to produce L1G digital images with the following grid cell characteristics:										
3.3.3.3.1	Grid cell size is variable from 15M to 60M in 0.001M increments				X						
3.3.3.3.2	Grid cell size is independently variable between spectral bands				X						
3.3.3.4	The LPGS shall produce L1G digital images that are spatially continuous between contiguous partial subintervals or WRS scenes.				X						
3.3.3.5	The LPGS shall have the capability to generate L1G digital images oriented by the following:				X						

## REVIEW

Rqmt. No.	Requirement Summary	PCS	DMS	RPS	GPS	QAS	AAS	UI	GL	HW	OPS
3.3.3.5.1	Oriented to Nominal Path				X						
3.3.3.5.2	Oriented to North Up				X						
3.3.3.6	The LPGS shall be capable of producing L1G data from L0R data for both the ascending and descending portions of the Landsat 7 orbit.				X						
3.3.3.7	The LPGS shall be able to produce L1G digital images for any combination of the eight spectral channels.				X						
3.3.3.8	The LPGS shall assemble and append to the L1G digital images all of the applicable metadata, quality and accounting data gathered in the construction of the L1G digital image. The complete L1G digital image package contains the following data elements as a minimum:		X								
3.3.3.8.1	Level 1G digital image (all requested bands)		X								
3.3.3.8.2	L1 metadata file		X								
3.3.3.8.3	DELETED										
3.3.4	Generate L1 metadata file.										
3.3.4.1	The LPGS shall generate ancillary L1R digital image data that describes the contents, processing parameters, and quality indicators of the L1R digital image.		X	X							
3.3.4.2	The LPGS shall generate ancillary L1G digital image that describes the contents, processing parameters, and quality indicators of the L1G digital image.		X		X						
3.3.4.3	The LPGS shall generate and append processing summary indicators specifying the algorithms applied to the Level 1 digital images.		X								
3.3.5	Assess L1 product quality.										

## REVIEW

Rqmt. No.	Requirement Summary	PCS	DMS	RPS	GPS	QAS	AAS	UI	GL	HW	OPS
3.3.5.1	The LPGS shall support automatic assessment of L1 digital image quality.					X					
3.3.5.2	The LPGS shall be able to optionally display any single band of the L1R digital image for visual quality assessment.					X	X	X		X	X
3.3.5.3	The LPGS shall be able to optionally display any single band of the L1G digital image for visual quality assessment					X	X	X		X	X
3.3.5.4	The LPGS shall be able to optionally print a color hardcopy of the display of any band(s) of the L1R digital image for visual quality assessment.					X	X	X		X	X
3.3.5.5	The LPGS shall be able to optionally print a color hardcopy of the display of any band(s) of the L1G digital image for visual quality assessment.					X	X	X		X	X
3.3.6	Transfer L1 file(s).										
3.3.6.1	The LPGS shall be able to output L1 digital images in the following formats:										
3.3.6.1.1	HDF-EOS (L1R and L1G)		X								
3.3.6.1.2	EOSAT FAST-Format (L1G only)		X								
3.3.6.1.3	GeoTIFF (L1G only)		X								
3.3.6.2	The LPGS shall transfer L1 files to ECS per the ECS to LPGS ICD.		X								
3.3.6.3	The LPGS shall provide the capability to display LPGS Level 1 file transfer summary upon operator request.		X					X		X	X
3.3.6.4	The LPGS shall be able to detect files that have been successfully transferred.		X								
3.3.6.5	The LPGS shall be able to mark successfully transferred files as candidates for deletion from LPGS temporary storage.		X								
3.3.7	Data storage										

## REVIEW

Rqmt. No.	Requirement Summary	PCS	DMS	RPS	GPS	QAS	AAS	UI	GL	HW	OPS
3.3.7.1	The LPGS shall be able to provide temporary online storage for the equivalent of 3 days of completed products. <b>NOTE:</b> CCR submitted to reword this requirement		X							X	
3.3.7.2	The LPGS shall be able retransmit files located in temporary storage.		X							X	
3.3.7.3	The LPGS shall be able to store Level 1 processing information on line for 90 days.									X	
3.3.7.4	The LPGS shall be able to transfer Level 1 processing information to offline storage after 90 days.		X							X	X
3.3.7.5	The LPGS shall be able to recover, display, and print Level 1 processing information located on offline storage for the life of the mission.		X					X		X	X
3.3.8	Control LPGS operations.										
3.3.8.1	The LPGS shall allow the operator to select thresholds for statistics and errors reported by the LPGS.							X			X
3.3.8.2	The LPGS shall automatically generate messages and alarms to alert the operator of LPGS results and errors exceeding operator selected thresholds.	X	X	X	X	X		X	X		X
3.3.8.3	The LPGS shall generate intermediate processing summaries on a periodic basis according to operator specification.	X		X	X			X		X	X
3.3.8.4	The LPGS shall provide an option to display L1 digital image quality status and statistics at operator request.	X				X	X	X		X	X
3.3.8.5	The LPGS shall provide an option to print L1 digital image quality status and statistics at operator request.	X				X	X	X		X	X

## REVIEW

Rqmt. No.	Requirement Summary	PCS	DMS	RPS	GPS	QAS	AAS	UI	GL	HW	OPS
3.3.8.6	The LPGS shall provide the capability to manually override the LPGS automated processing functions.							X			X
3.3.8.7	The LPGS shall provide the manual capability to cancel Level 1 processing prior to completion of digital image generation.	X						X			X
3.3.8.8	The LPGS shall be able to display and print trouble tickets received from ECS.						X	X		X	X
4	LPGS Performance Requirements										
4.1	Performance Requirements.										
4.1.1	The LPGS shall be capable of processing a volume of data equivalent to 28 ( accounts for 10 percent of LPGS internal reprocessing) standard L0R WRS scenes to Level 1 digital images each day.	X	X	X	X	X				X	
4.1.2	The LPGS shall contribute no greater than .7 percent uncertainty to absolute radiometric accuracy during the generation of L1R and 1G digital images.			X	X						
4.1.3	The LPGS shall contribute circular errors no greater than 1.8 m, 1 sigma, in the production of systematically corrected L1G digital images.				X						
4.1.4	The LPGS shall provide at least 110 percent of the processing throughput capability required to satisfy the worst case processor loading.									X	
4.1.5	The LPGS shall provide at least 125 percent of the random access memory capacity required to satisfy the worst case memory loading.									X	

## REVIEW

Rqmt. No.	Requirement Summary	PCS	DMS	RPS	GPS	QAS	AAS	UI	GL	HW	OPS
4.1.6	The LPGS shall provide at least 125 percent of the peripheral storage capacity required to satisfy the worst case peripheral storage loading.									X	
4.1.7	Deleted										
4.1.8	The LPGS shall produce Level 1G products that are accurate to within 250 meters cross track and 250 meters along track using geometric calibration information generated by IAS and contained in the associated calibration parameter file.				X						
4.1.9	<b>NOTE:</b> This is a new requirement to be added. Register pixels from one spectral band to the...				X						
4.2	External Interface Performance Requirements										
4.2.1	The LPGS shall be able to ingest from ECS a data volume equivalent to 3 WRS scenes' worth of standard LOR data for each Level 1 digital image request.		X							X	
4.2.2	The LPGS shall have the capability to support the transfer to ECS of the equivalent of a minimum of 25 WRS sized Level 1 digital images per day.		X							X	
4.2.3	The LPGS-ECS interface shall provide the capability to transfer to the ECS at least 33 GB of Level 1 output files per day.		X							X	
4.3	Reliability, maintainability, and availability.										
4.3.1	The LPGS shall provide an operational availability of 0.96 (TBR).	X	X	X	X	X	X			X	
4.3.2	The LPGS shall support a mean time to restore (MTTR) capability of 4 hours (TBR).	X	X	X	X	X	X			X	

## REVIEW

Rqmt. No.	Requirement Summary	PCS	DMS	RPS	GPS	QAS	AAS	UI	GL	HW	OPS
4.4	Security.										
4.4.1	The LPGS shall provide system, network, and operations security according to the ESDIS security policy (Applicable Document 8) and the NASA AIS Handbook (Applicable Document 9).	X	X	X	X	X	X			X	X

## Appendix B. LPGS Software Size Estimates

### B.1 LPGS Estimated Lines of Code

Table B-1 presents delivered source instructions (DSI) estimates for LPGS software based on the detailed design of six LPGS subsystems and software modules including the database and global routines. The DSI estimate is based on approximately 100 DSI per module, with the exception of user interface- and database-related modules. Each user interface screen is estimated at 200 DSI. Each database related module is sized at approximately 50 DSI. For the most part, each LPGS task invokes a subset of the global and database modules.

**Table B-1. DSI Estimates for LPGS Software at Detailed Design**

LPGS Software Subsystem	No. of Tasks	No of Modules	Estimated DSI
Process Control Subsystem (PCS)	4	24	2,700
Data Management Subsystem (DMS)	7	128	12,130
Radiometric Processing Subsystem (RPS)	–	–	24,000
Geometric Processing Subsystem (GPS)	–	–	39,000
Quality Assessment Subsystem (QAS)	3	11	2,400
Anomaly Analysis Subsystem (AAS)	4	6	2,500
User Interface (UI)	–	40	8,000
Database (DB)	–	192	10,235
Global Modules	–	53	4,895
Test/Diagnostics	–	-	3,500
<b>Total</b>			<b>109,360</b>

## Appendix C. Database Table Definition Report

---

TBS

## REVIEW

### Abbreviations and Acronyms

---

AA	anomaly analysis
AAS	Anomaly Analysis Subsystem
AAUI	AAS analyst user interface
AIT	Algorithm Implementation Team
API	applications programmatic interface
ASCII	American Standard Code for Information Interchange
AUI	analyst user interface
CASE	computer-aided software engineering
CD-ROM	compact disc read-only memory
CCR	configuration change request
CDE	Oracle Corporation's Cooperative Development Environment
CDS	critical design specification
CNMOS	Consolidated Network and Mission Operations Support
COTS	commercial off-the-shelf
CPF	calibration parameter file
CPU	central processing unit
DAA	data availability acknowledgment
DAAC	Distributed Active Archive Center
DAN	data availability notice
DAT	digital audio tape
DBAR	Database Access Routine
DBMS	Database Management System
DD	data dictionary
DDA	data delivery acknowledgment
DDE	data dictionary entry
DDN	data delivery notice
DFCB	data format control book

## REVIEW

DFD	data flow diagram
DFL	DMS Format L1 Product
DGR	DMS Generate Reports
DHF	Data Handling Facility
DIE	DMS IF With ECS
DIL	DMS Ingest LOR Product
DMS	Data Management Subsystem
DPL	DMS Process LOR Product
DRM	DMS Resource Manager
DSI	delivered source instructions
DSS	data server subsystem
DXL	DMS Xmit L1 Product
E&A	evaluation and analysis
ECS	EOSDIS Core System
EDC	EROS Data Center
EGS	EOS Ground System
EOS	Earth Observing System
EOSAT	Earth Observation Satellite Company
EOSDIS	EOS Data and Information System
ERD	entity relationship diagram
EROS	Earth Resources Observation System
ESDIS	Earth Science Data and Information System
ETM+	Enhanced Thematic Mapper Plus
F&PRS	functional and performance requirements specification
FAST	Fast Argonne System for Transport; an output format for L1 digital images
FDDI	fiber-optic data distribution interface
FIFO	first in, first out
ftp	file transfer protocol
GB	gigabyte

## REVIEW

GCP	ground control point
GDS	ground data system
GeoTIFF	Geographic Tag(ged) Image File Format; an output format for L1 digital images
GPS	Geometric Processing Subsystem
GSFC	Goddard Space Flight Center
GUI	graphical user interface
HDF	Hierarchical Data Format
HWC	hardware component
HWCI	hardware configuration item
I&T	integration and test
I/O	input/output
IAS	Image Assessment System
IC	internal calibrator
ICD	interface control document
IDD	interface data descriptions
IDL	Interactive Data Language
IGS	international ground station
IPC	interprocess communication
IRD	interface requirements document
ISO	International Standards Organization
LAN	local area network
L0R	Level 0R
L1	Level 1
L1G	Level 1 geometrically corrected
L1R	Level 1 radiometrically corrected
Landsat	Land Satellite
LGN	Landsat ground network
LGS	Landsat 7 ground station
LPGS	Level 1 Product Generation System

## REVIEW

LPS	Landsat 7 Processing System
M	meter
MB	megabyte
MBps	megabytes per second
MIPS	million instructions per second
mm	millimeter
MMO	Mission Management Office
MO&DSD	Mission Operations and Data Systems Directorate
MO&SDD	Mission Operations and Systems Development Division
MOC	Mission Operations Center
MSCD	mirror scan correction data
M-Specs	module specifications
MSS	management subsystem
MTF	modulation transfer function
MTPE	Mission to Planet Earth
MTTR	mean time to restore
NASA	National Aeronautics and Space Administration
NCSA	National Center for Supercomputing Applications
NFS	Network File System
NOAA	National Oceanic and Atmospheric Administration
ODL	Object Descriptive Language
OUI	operator user interface
PAN	product acceptance notice
PC	personal computer
PCD	payload correction data
PCMB	Project Configuration Management Board
PCS	Process Control Subsystem
PDL	program design language
PDR	product delivery record
PDRD	product delivery record discrepancy

## REVIEW

POSIX	portable operating system interface for UNIX
PRQ	PCS Request Processor
PSI	PCS System Initialization/Termination
PSO	Project Science Office
P-Specs	process specifications
PWC	PCS Work Order Controller
PWG	PCS Work Order Generator
PWS	PCS Work Order Scheduler
Q1G	L1G Quality Assessment (task)
Q1R	L1R Quality Assessment (task)
QA	quality assessment
QAS	Quality Assessment Subsystem
QUI	Quality Assessment User Interface (task)
RAID	redundant array of inexpensive devices
RAM	random access memory
RMA	reliability, maintainability, and availability
RPC	remote procedure call
RPS	Radiometric Processing Subsystem
RSI	Research Systems, Inc.
RTM	Requirements and Traceability Management (tool)
SAT	Shift Along Track
SCSI	multiple small computer system interface
SDPS	science data processing segment
SDR	system design review
SDS	system design specification
SEAS	Systems, Engineering, and Analysis Support
SGI	Silicon Graphics, Inc.
SNR	signal-to-noise ratio
SQL	Structured Query Language
SRR	system requirements review

## REVIEW

SSDM	SEAS System Development Methodology
SSR	solid-state recorder
SWCI	software configuration item
TBD	to be determined
TBR	to be resolved
TBS	to be supplied
TIFF	tagged image file format
UI	user interface
URF	user request file
USGS	United States Geological Survey
UTM	Universal Transverse Mercator
VME	Versa Module European
WO	work order
WRS	Worldwide Reference System